МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний економічний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерних наук

КОМАР Дмитро Вікторович

Додаток для перегляду оцінок студентів ТНЕУ для IOS/ IOS application for viewing marks of TNEU students

напрям підготовки: 6.050103 - Програмна інженерія
фахове спрямування - Програмне забезпечення систем

Бакалаврська дипломна робота

Виконав студент групи ПЗС-41
Д. В. Комар
_____

Науковий керівник:
к.т.н., доцент СПІЛЬЧУК В.М.
_____

Бакалаврську дипломну роботу
допущено до захисту:

"___" _____ 20___ р.

Завідувач кафедри
_____ **А. В. Пукас**

ТЕРНОПІЛЬ - 2016

ЗМІСТ

ВСТУП

У сучасних умовах швидкий доступ до інформації інколи має ключове значення, тому кожна сфера повинна мати найшвидші та найзручніші сервіси.

У кожній організаційній структурі є низка проблем. Система оцінок не є виключенням. В даний час в університетах дуже популярні системи моніторингу оцінок. Всі ці системи об'єднує одна проблема – відсутність додатків на популярних платформах. На даний момент, всі свої оцінки можна побачити лише у веб-версії системи.

Зважаючи на ріст ринку мобільних додатків та смартфонів, система оцінок ТНЕУ повинна бути доступна як повноцінний мобільний додаток, а зважаючи на те що iOS є домінуючою мобільною платформою розробка під неї є пріоритетною на мобільному у ринку.

З впровадженням автоматизованої системи доступ до інформації дуже сильно спростяться. Багато дій які вимагали уваги і забирали багато часу будуть автоматизовані. Буде реалізовано збереження паролю, кешування сесії, зручний інтерфейс. Все це спростить процес і позбавить студентів від непотрібної роботи.

Метою даної дипломної роботи є виконання всіх етапів розробки програмного забезпечення та дотримання усіх норм та стандартів. Кінцевим результатом роботи буде вузькопрофільна програмна система перегляду оцінок для студентів ТНЕУ.

На всіх етапах розробки системи було використано низку інструментів, а саме:

- Microsoft Visio 2012, що використовується з метою розробки різних видів діаграм та схем, які представлені в розділі аналізу предметної області та проектування системи;

- Xcode 7.3. Середовище з використання об’єктно-орієнтованого підходу для розробки програмного забезпечення та додатків із графічним інтерфейсом;
- Atlassian SourceTree. Система для роботи з .git репозиторієм ;

Дипломна робота надає можливість закріпити та удосконалити теоретичні та практичні навички. Розроблена програмна система, при затвердженні її керівництвом організації може бути впроваджена та застосована.

РОЗДІЛ 1

АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ СИСТЕМИ ПЕРЕГЛЯДУ ОЦІНОК

1.1 Коротка характеристика об’єкту управління системи перегляду оцінок

Об’єктом дослідження дипломної роботи є система перегляду оцінок для студентів ТНЕУ. Система призначена для перегляду модульних оцінок з предметів, що викладались в поточному семестрі.

Щоб почати користування системою необхідна реєстрація.

Процес Реєстрації в системі складається з двох частин.

1. На сторінці реєстрації потрібно ввести:

- Текст логіну (довжиною 4 - 16 знаків, можна використовувати символи: a-z, 0-9, тире та крапку)

- Текст паролю (довжиною 4 - 16 знаків)

- email (на неї буде відправлено лист)

- # залікової книжки

Після вдалої реєстрації на email буде відправлено листа.

2. Після підтвердження потрібно звернутись у деканат щою підтвердити реєстрацію у системі

Сторінка авторизації (вигляд з телефону) зображена на рисуноку 1.1

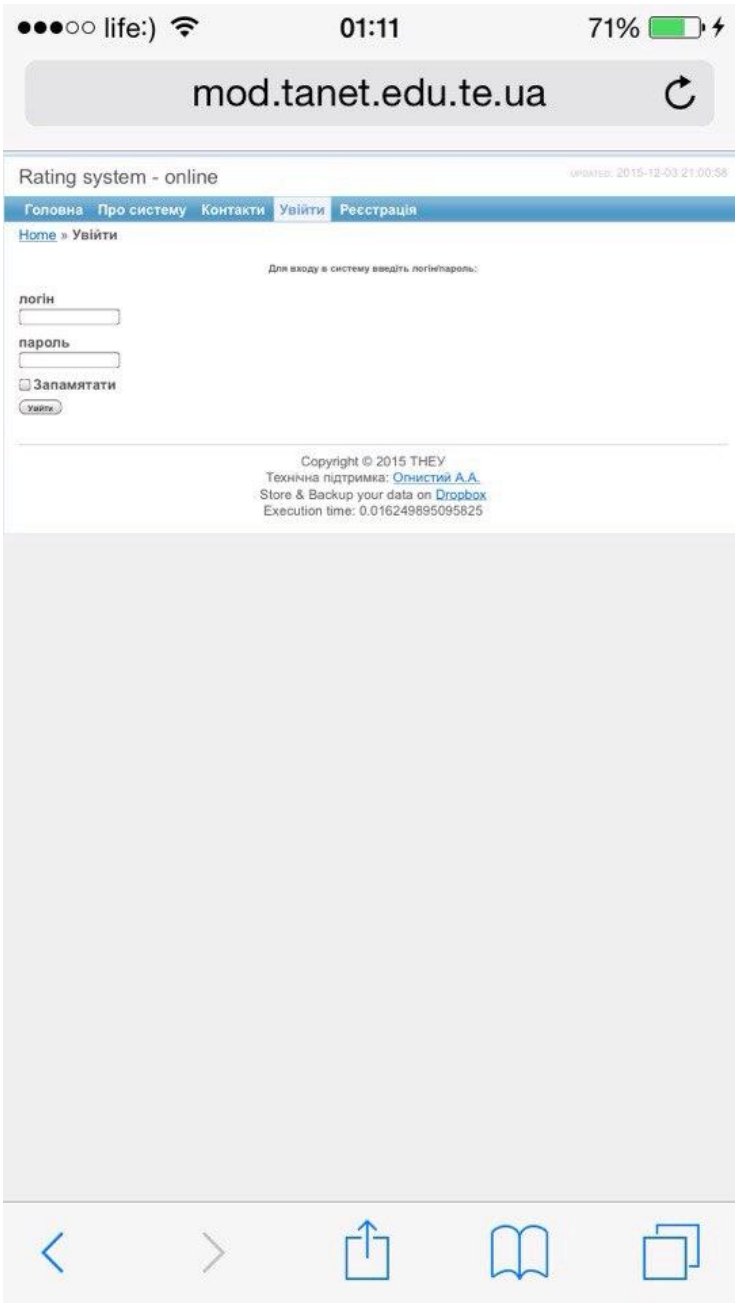


Рисунок 1.1 – Сторінка авторизації (вигляд з телефону)

На сьогоднішній день існує сайт http://mod.tanet.edu.te.ua який не оптимізований для мобільного телефона і шоб переглянути будь яку інформацію про модулі студенту потрібно виконувати не зручний алгоритм дій, перед тим як отримати інформацію.

Рисунок 1.2 – Сторінка перегляду оцінок (вигляд з телефону)

1.2 Опис предметної області

Предметною областю реалізованої задачі є така інформаційна сутність як система перегляду оцінок, що включає в себе інформаційні потоки про оцінки студентів, дати модулів та їх вагу.

Студент – це єдиний тип користовачів системи який може зробити вхід при наявності пароля та логіну та побачити свою успішність.

У наш час провідні університети мають системи перегляду оцінок для своїх студентів, що спрощує режим навчання, тому розробка системи для iOS є пріоритетною.

У бізнес-процесі «Вхід в систему» зображено проходження входу в систему оцінок.

Схему бізнес процесу приведено у додатку А.

У таблиці 1.1 наведена коротка характеристика бізнес-процесу « Вхід в систему ». Вхідними документами вважаються логін (ім’я користувача) та пароль тобто, авторизація.

Таблиця 1.1

Характеристика бізнес-процесу « Вхід в систему »

Назва характеристики	Значення характеристики
Ім’я бізнес-процесу	Вхід в систему
Основні учасники	Студент
Вхідна подія	Вхід
Вхідні документи	Логін, пароль
Вихідна подія	-
Вихідні документи	-

1.3 Огляд і аналіз існуючих аналогів, що реалізують функції предметної області

Щоб створити унікальне, високонадійне, ефективне, з багатьма можливостями програмне забезпечення проведено порівняння вже існуючих аналогів програмних продуктів для певної області застосування. Для цього було взято відоме програмне забезпечення ModuleOK та Аналитика Оценок.

Таблиця 1.2

Порівняльння програмних продуктів

Назва програмного продукту	ModuleOK	Аналитика Оценок
Фірма-розробник	Україна, тернопіль	Росія, Москва
Функціональність	Програмний продукт дозволяє: 1) Перегляд оцінок; 2) Перегляд графіку успішності 3) Перегляд статистики 4) Перегляд стрінки Вконтакті 5) Перегляд дат модулів	Програмний продукт вміщує: 1) Перегляд оцінок;

Продовження Таблиці 1.2

Інтерфейс користувача	Інтерфейс даної програми не зовсім зручний. Мова програми – українська. Система недоступна для iOS та OS X	Інтерфейс програми є зручним у використанні. У головному вікні програми висвітлені всі пункти меню роботи. Система доступна тільки для Росії
Допомога корисчу	В даному програмному продукті розробник не надав жодної допомоги користувачеві.	В головного меню програми є пункт «Помощь», де описано інформацію про програму, версію та контактні дані для додаткових питань стосовно використання. Також в додатку до програми входить презентація покрокової роботи з програмою.

Розглянемо детальніше роботу програми « ModuleOK ».

На рисунку 1.3 зображено вікно входу в систему. За допомогою вже внесених даних, відбувається вхід. Нажаль мені не вдалося дістатись далі меню логіну через те що система має баг і логін та пароль не ввдодяться. Розробник повідомив мені що існує баг із

браузером Сафарі, який використовується у iOS та OS X усіх версій та на всіх девайсах.

Шляхом зміни девайсу мені все ж таки вдалось побачити дизан додатку який зображено на рисунку 1.4

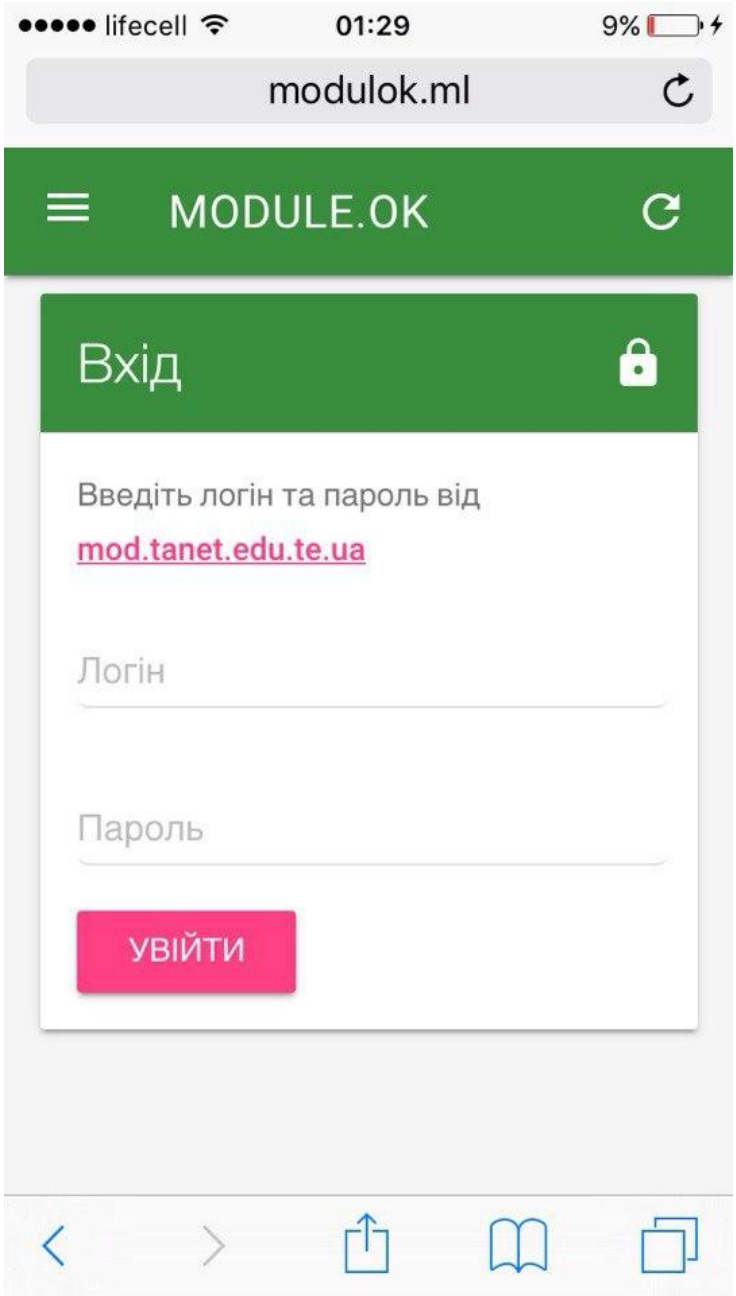


Рисунок 1.3 – Екран входу

Рисунок 1.4 – Дизайн додатку

Розглянемо детальніше роботу програми « Аналитика Оценок ».

Головне вікно програмної системи містить меню опцій даної програми, яке відображається зліва. На рисунку 1.5 зображено вікно входу в систему.

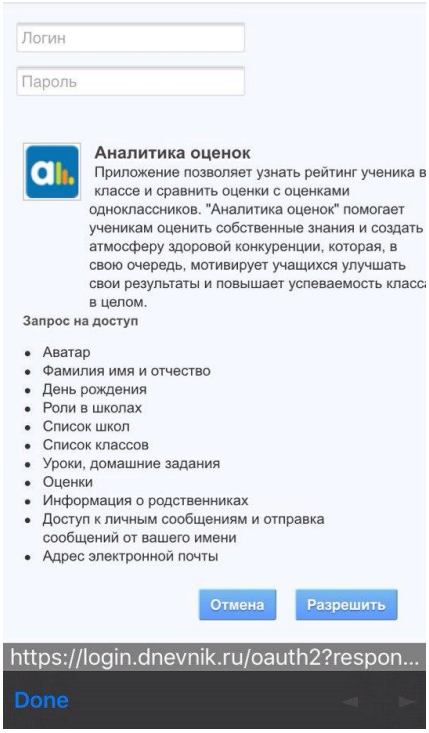


Рисунок 1.5 – Вікно входу в систему

Дизайн системи зображено на рисунку 1.6

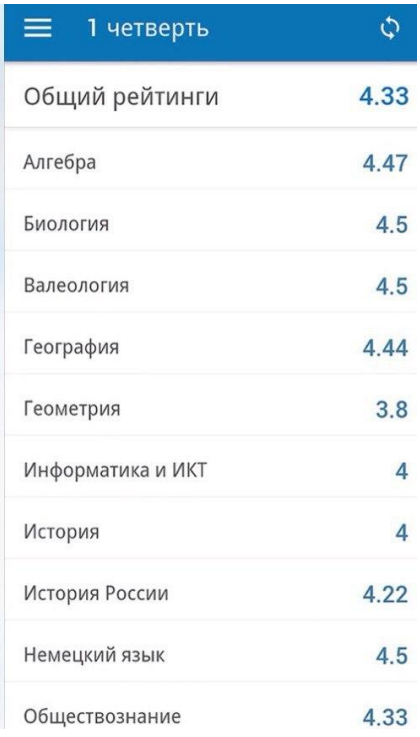


Рисунок 1.6 – Дизайн системи

1.4 Специфікація вимог до системи

Введемо деякий список понять в області розробки ПЗ так званий глосарій. Глосарій – словник до тексту, що пояснює маловідомі або застарілі слова, це список понять в специфічній області знань з їх визначеннями. Ці поняття визначень описані у додатку В.

Візуальне моделювання системи можна уявити, як певний процес поетапного спуску від найбільш загальної та абстрактної концептуальної моделі вихідної системи до логічної, а потім і до фізичної моделі відповідної програмної системи.

Діаграма прецедентів дозволяє відобразити список операцій, виконуваних системою.

Кожна діаграма або, як її зазвичай називають, кожен Use case - це опис сценарію поведінки, якій слідують актои (дійові особи)

Use-case відображає об'єкти як системи, так і предметної області і задачі, ними виконувані.

Розроблювана система повинна мати такі функції:

- авторизація користувача в системі;

Функція *«Авторизація користувача в системі»*. Для того щоб зайти в свій аккаунт в додатку користувач повинен заповнити форму авторизації, а саме два її поля: «Логін», «Пароль». Після цього авторизований користувач буде мати доступ до свого кабінету.

Діаграма прецедентів показана на рисунку 1.7.

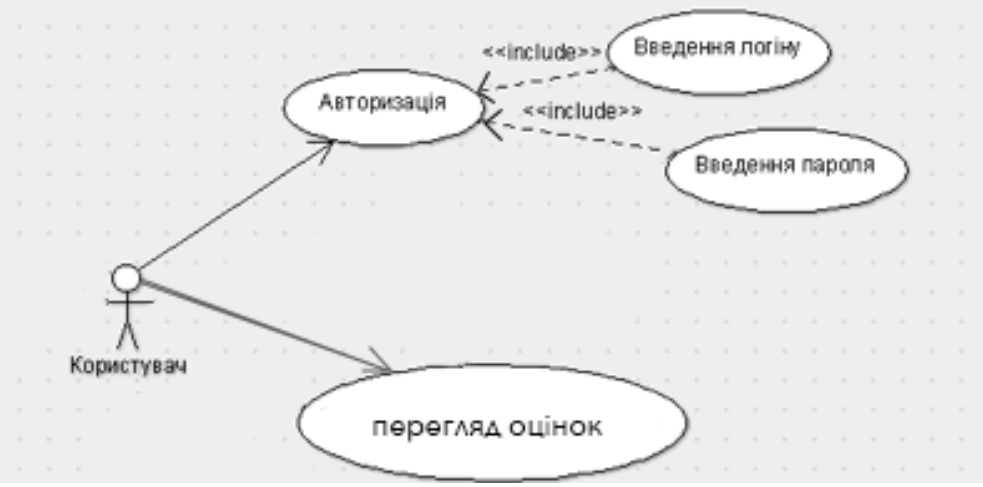


Рисунок.1.7 – Діаграма прецедентів

Діаграма Прецедентів відбражає варіанти використання системи та можливості користувача (актора) у системі, будується на основі UML діаграми та зовражає всі ті самі процеси із позиції користувача.

Кожен варіант використання зображає певний набір дій. При цьому алгоритм виконання цих дів не вказано

Діаграма підтримує “включення”, що дає вожливість розширювати функціонал певних варіантів використання спеціальними випадками, при цьому змін/ється основний елемент.

Цей тип діаграми не акцентує увагу на конкретній взаємодії, головний акцент приділяється на процесі авторизіції за допомогою логіну та паролю які студен може отпримати в деканаті, попередньо зареєструвавшись на сайті рейтингової системи. А також на процесі перегляді оцінок, модулів, дат модулів, ваги модулів, назв дисціплін та загального підсумкового балу із дислипліни.

Діаграма послідовності для варіанту використання «Авторизація» зображена на рисунку 1.8.

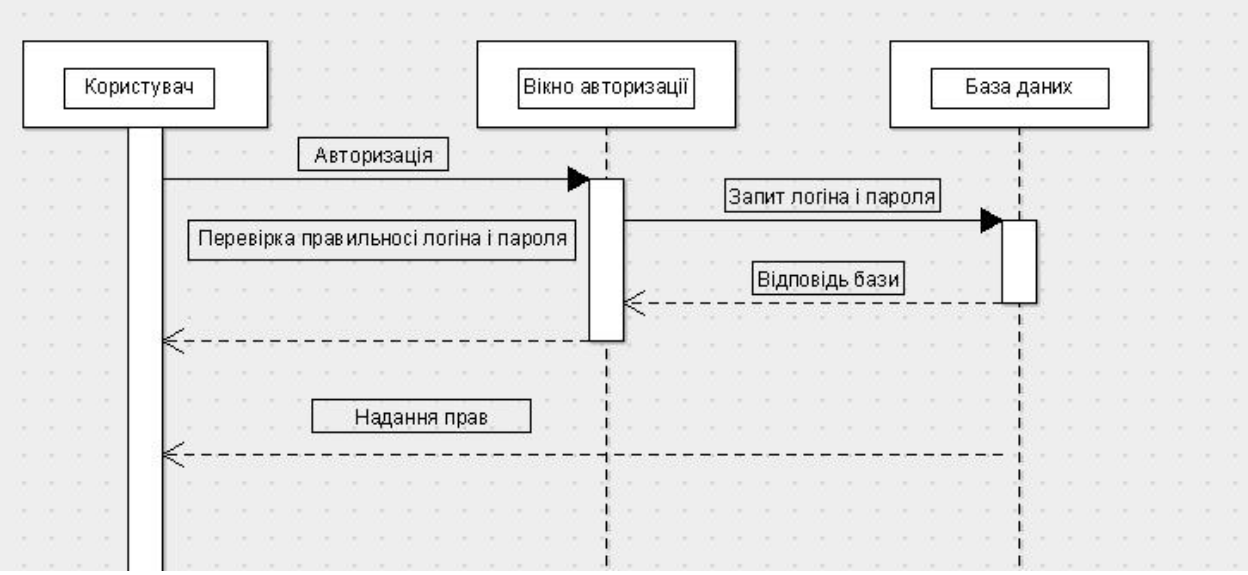


Рисунок.1.8 – діаграма послідовності для прецеденту «Авторизація».

Діаграма активності є розвитком діаграми станів. Цей тип діаграм використовується для перегляду станів модельованого об'єкту, проте, головне призначення Activity diagram в тому, щоб показувати бізнес-процеси об'єкту. Цей тип діаграм дозволяє розпаралелення і взаємну та взаємно пропорційну синхронізацію процесів, незалежно від їх складності.

Такий вид діаграм дозволяє проектувати алгоритми поведінки об'єктів будь-якої складності, зокрема може використовуватися для проектування блок-схем. Цей тип діаграм поширений у багатьох методологіях розробки програмного забезпечення навіть у теперішній час, тому що є класичним прикладом зображення процесів та взаємодії між ними. Має інтуітивно зрозуміле зображення, що дає можливість людям не знайомим із правилами запису орієнтуватись, що він описує.

Діаграма активності для прецендету «Авторизація» зображена на рисунку 1.9.

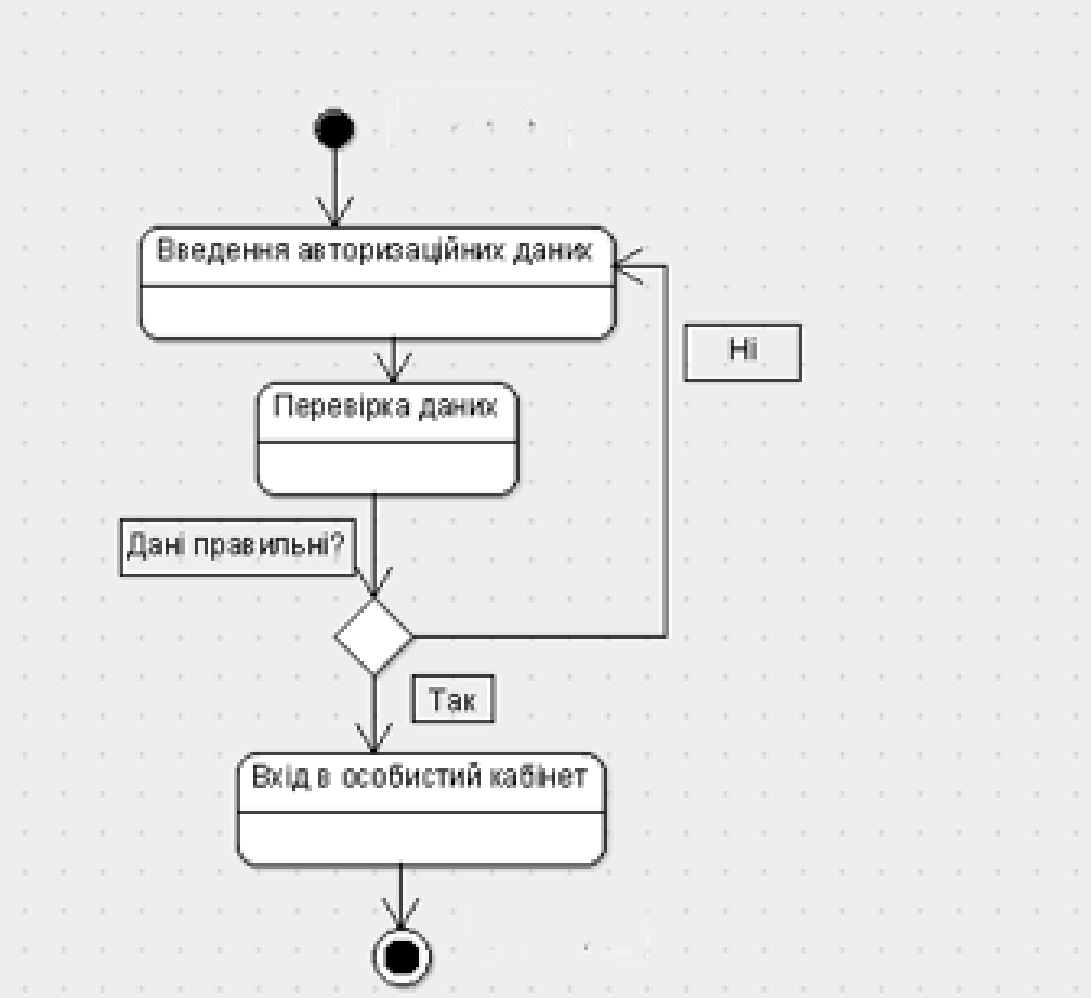


Рисунок.1.9 – Діаграма активності для прецеденту «Авторизація»

Діаграми компонент призначені для розподілу класів та об’єктів по компонентах при фізичному проектуванні системи. Часто даний тип діаграм називають діаграмами модулів.

При проектуванні великих систем може виявитися, що систему необхідно розкласти на декілька сотень чи тисяч компонентів і такий тип діаграми дозволяє систематизувати велику кількість модулів та зв’язків між ними.

Діаграма компонентів системи зображена на рисунку 1.10.

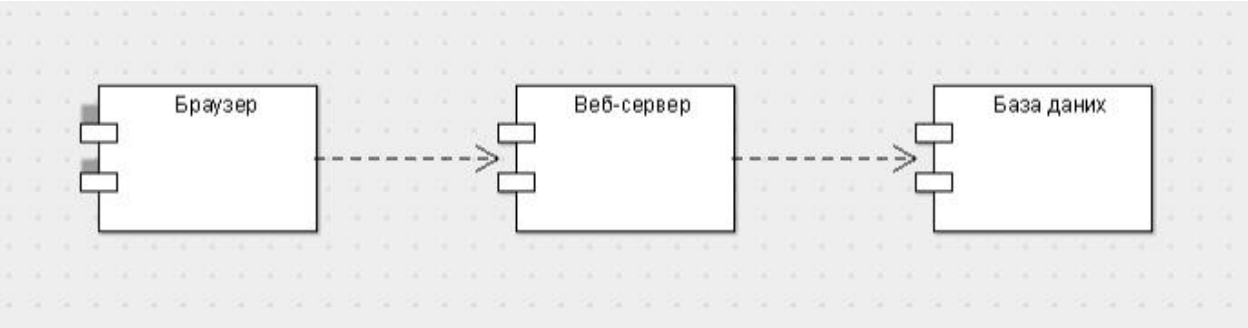


Рисунок.1.10 – Діаграма потоків даних для прецденту «Авторизація»

Представимо варіанти використання по кадрам, тобто ескізам екранних форм. Зобразимо декілька основних екранних форм використання.

На рисунку 1.11 зображено майбутній вигляд функції логіну в систему.

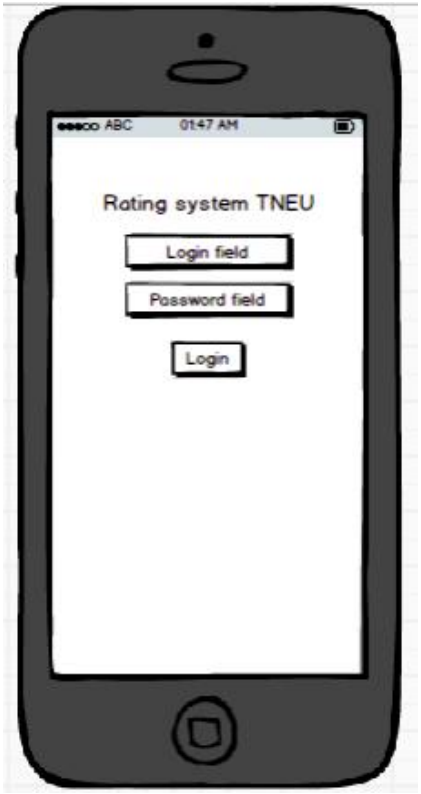


Рисунок 1.11 – Екран входу в систему

Ескіз, що зображений на рисунку 1.12 представляє перегляд оцінок у системі

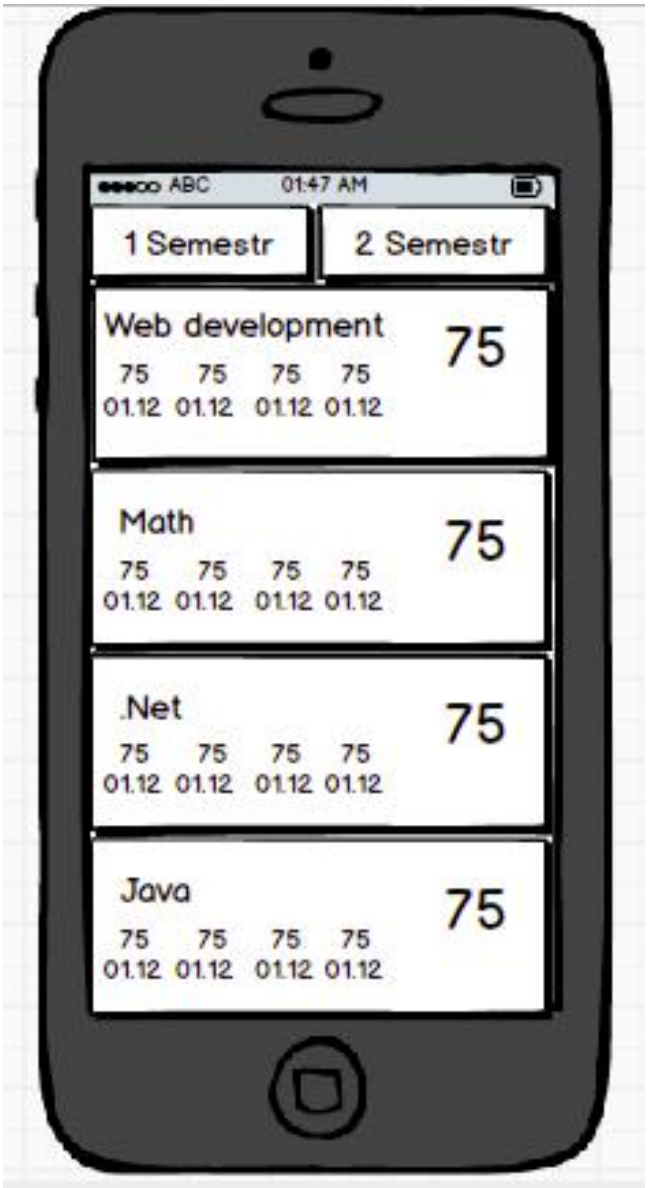


Рисунок 1.12 – Екран перегляду оцінок

Специфікацію функціональних вимог програмної системи описано у таблиці 1.6. Специфікацію нефункціональних вимог програмної системи описано в таблиці 1.7.

Таблиця 1.3

Специфікація функціональних вимог

Ідентифікатор вимоги	Назва вимоги варіанту використання	Атрибути вимог		
		Пріоритет	Складність	Контакт
1.	Реєстрація користувача	Обов’язкова	Низька	Інженер
2.	Авторизація користувача	Обов’язкова	Низька	Інженер
3.	Додавання нового матеріалу	Обов’язкова	Низька	Бухгалтер

Таблиця 1.4

Специфікація нефункціональних вимог

Ідентифікатор вимоги	Назва вимоги	Атрибути вимог		
		Пріоритет	Складність	Контакт
1. Застосовність				
1.1	Час,для навчання звичайних користувачів	Рекомендована	Низька	Інженер

Продовження Таблиці 1.4

1.2	Основні вимоги застосовності нової системи відносно інших систем, які знають користувачі	Опційна	Низька	Інженер
1.3	Вимоги по відповідальності стандартам графічного інтерфейсу	Рекомендована	Низька	Інженер
1. Надійність				
2.1	Доступність	Обов’язкова	Середня	Інженер
2.2	Середній час безвідмовної роботи	Рекомендована	Середня	Інженер
2.3	Точність	Обов’язкова	Середня	Інженер
3. Робочі характеристики				
3.1	Використання ресурсів	Рекомендована	Середня	Інженер
4. Проектні обмеження				
4.1	Вимоги до технології програмування	Рекомендована	Середня	Інженер

Значення нефункціональних вимог:

- Час, необхідний для навчання звичайних і досвідчених користувачів становить – 5 хвилин (звичайні користувачі), 3 хвилини (досвідчені);

- Основні вимоги застосовності нової системи відносно інших систем, які знають користувачі – система має зручне розміщення всіх елементів, що не призведе до ускладнення роботи користувачів;
- Вимоги по відповідальності стандартам графічного інтерфейсу користувача – використоруються стандарти графічного інтерфейсу iOS
- Доступність. Повинен бути не менше 95% доступного часу;
- Середній час безвідмовності роботи становить 1 місяць;
- Використання ресурсів. Для роботи із програмою повиннно бути не менше 5МБ дискового простору, iOS 8.0 або вище
- Вимоги до технології програмування. Система розроблена на Swift 2.1

Висновки до розділу 1

1. Проаналізовано предметну область системи перегляду оцінок, встановлено, які задачі потребують автоматизації.
2. Досліджено організаційну структуру системи оцінок.
3. Спроектовано мокапи для розробки системи.

4.Описано варіанти використання програмної системи та нефункціональні вимоги продукту.

РОЗДІЛ 2

ПРОЕКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

2.1 Розробка архітектури програмної системи

UML-діаграма класів реалізує основну бізнес-логіку програмної системи. В цій діаграмі зображено певні класи (об’єкти) і їхні зв’язки між собою. UML-діаграма класів наведена на рисунку 2.1.

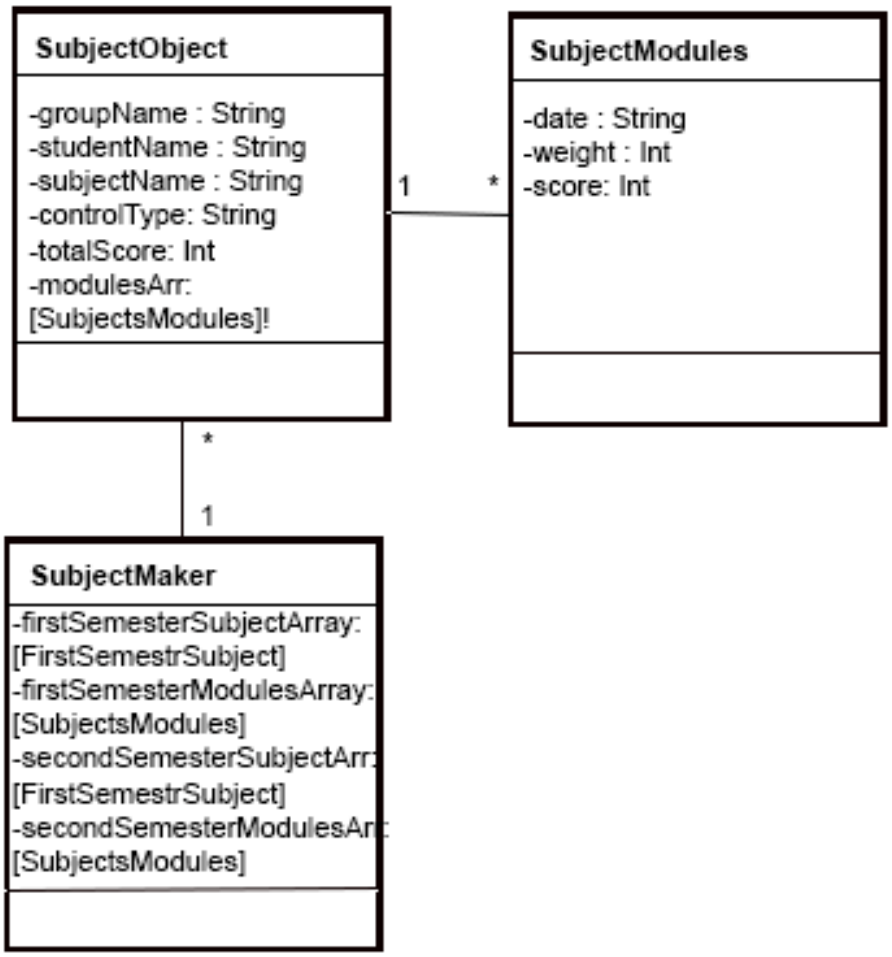


Рисунок 2.1 – Діаграма класів програми

Короткий опис і призначення кожного класу наведено у таблиці 2.1.

Таблиця 2.1

Характеристика класів

Клас	Призначення класу	Атрибути та їх значення
SemestrObject	Масив для створення об’єкту предмет	groupName – назва групи; studentName – ім’я студента; subjectName – назва предмету; controlType – тип контролю; totalScore – оцінка загальна; moduleType – масив модулів;
SubjectsModules	Масив для створення об’єкту модуль	date – дата weight – вага score - оцінка
SubjectMaker	Парсер JSONy	-

Діаграма станів представляє собою те, що кожен об'єкт системи, який володіє конкретною поведінкою, може знаходиться в конкретних станах, переходити із стану в стан, здійснюючи певні дії в процесі реалізації сценарію поведінки об'єкту. Поведінку об'єктів існуючих систем можна представити з погляду теорії кінцевих автоматів, що означає що поведінка об'єкту відбивається в його станах, і даний тип діаграм дозволяє відобразити це графічно.

Діаграма IDEF0 зображена на рисунку 2.2.

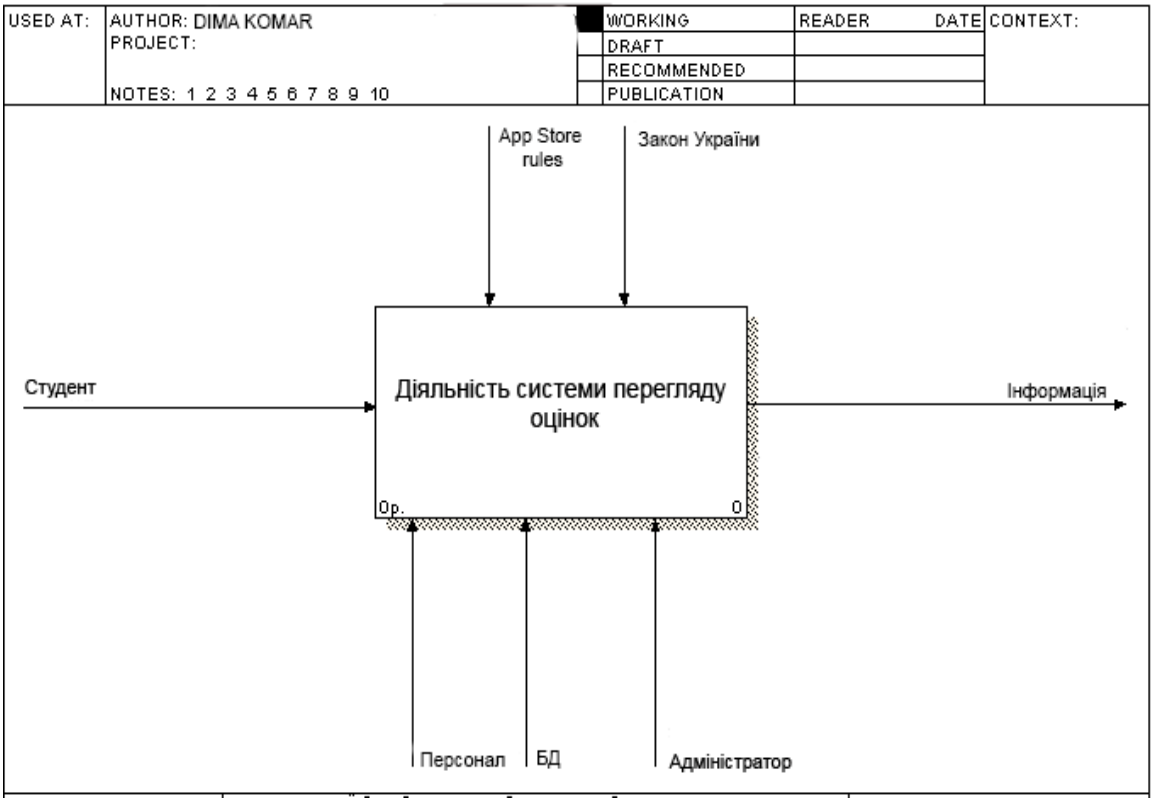


Рисунок.2.2 – Діаграма IDEF0

Діаграма DFD зображена на рисунку 2.3.

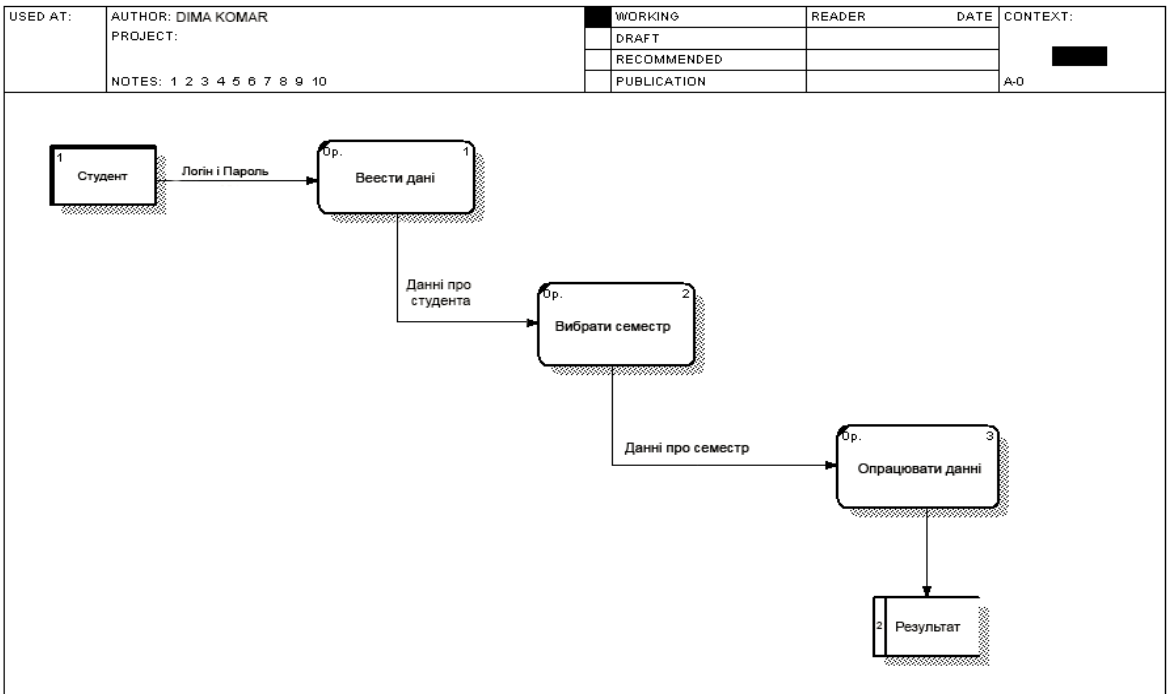


Рисунок.2.3 – Data flow diagram

Діаграма станів для для прецеденту «Авторизація» зображена на рисунку 2.4.

Рисунок.2.4 – Діаграма станів для для прецеденту «Авторизація»

Висновки до розділу 2

1. Розроблено архітектуру програмної системи та охарактеризовано класи із атрибутами, спроектовано відповідні зв’язки між ними.
2. Розглянуто поведінку системи при кожному задіяному варіанті використання з використанням діаграми станів.
3. За допомогою діаграми потоків даних створено модель функціональних вимог програмної системи перегляду оцінок.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Програмна реалізація проекту

iOS Додаток для перегляду оцінок студентів ТНЕУ був реалізований на платформі iOS із використанням мови Swift. Swift — мультипарадигмова компільована мова для розробки додатків.

Переваги мови Swift:

- Об’єктно-орієнтований підхід. Середовище Swift повністю базується на об’єктно-орієнтованих принципах;
- Можливість написання коду за правилами різних парадигм;
- Безпека;
- Швидкість написання коду
- Швидкість виконання коду

Одною із найголовніших переваг мови програмування Swift є спрямованість її на можливість повторного використання створених компонентів. Також потрібно відмітити таке:

- Swift є цілком об’єктно-орієнтованою мовою з можливостями спадкоємства, де навіть типи, вбудовані в мову, представлені класами;
- Простіша та більш надійна мова, ніж її попередники.
- Swift має вікритий код, тому кожен може робити її кращою

Інтерфейс користувача даної програмної системи розроблений за допомогою Storyboard та задовольняє основні вимоги, а зокрема: зручність та доступність у використанні. Виконання будь-якої функції не забере багато часу та зусиль користувача.

Для програмної системи рекомендуються такі вимого щодо апаратного забезпечення:

- iPhone 4s+
- Вільне місце на диску розміром не менше 50 Мб;
- Кредитна Картка

Система спроектована з використанням об’єктно-орієнтованого підходу, тобто розроблена у вигляді об’єктів (сукупності функціональних елементів). У об’єктно-орієнтованому програмуванні використовуються класи, які пов’язані між собою атрибутами і методами (функціями), що здійснюють операції над даними.

В якості специфікації модулів представлено опис класів, які були створені та використані для роботи програми. На рисунку 3.1 зображено діаграму класів.

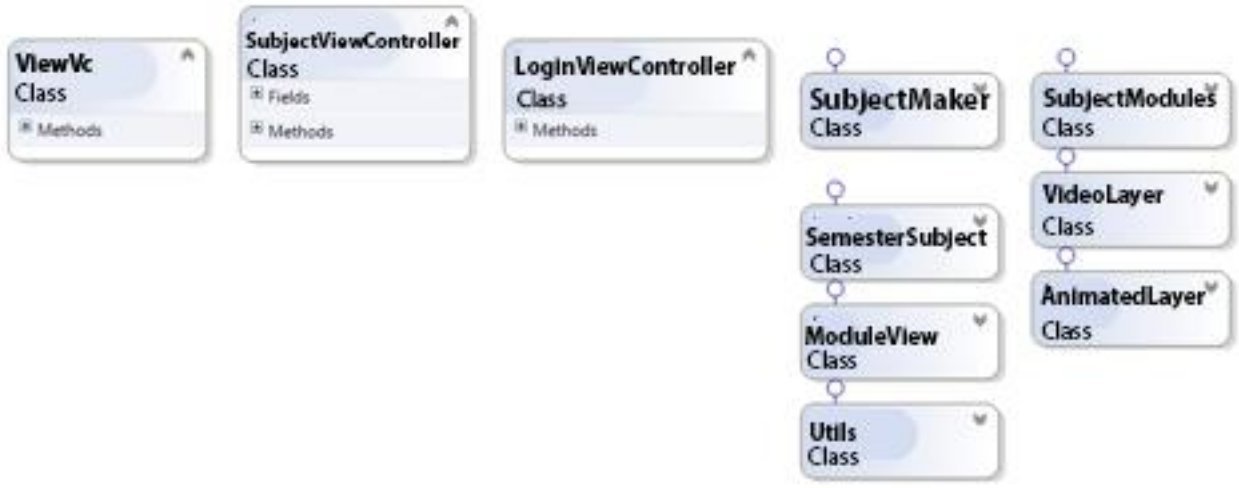


Рисунок 3.1 – Діаграма класів

Класи SemestrSubject, SubjectsModules є класами, які означають сутність. Вони містяться тільки атрибути. Опис інших класів

(LoginController, SubjectViewController, SubjectMaker, ModuleView, AnimatedView, VideoLayer, та) методами, що виконують важливі функції програми, розглянуто нижче.

Клас LoginController містить методи:

- viewDidLoad() – викликається перед завантаженням екрану;
- viewWillAppear () – викликається перед появою екрану;
- viewDidAppear() – викликається коли екран вже з’явився
- textFieldDidChange() – викликається коли користувач змінює текст
- loginButtonEnabled() – повертає стан кнопки логіну
- loginButtonPressed() – обробляє натиск кнопки логіну
- setupReachability() – робить перевірку на наявність мережі

Клас SubjectViewController містить методи:

- viewDidLoad() – викликається перед завантаженням екрану;
- viewWillAppear () – викликається перед появою екрану;
- viewDidAppear() – викликається коли екран вже з’явився
- switchChangedValue () – обробляє зміну свіча зверху екрану;
- getModulesOutOfSemestr() - визначає логіку сортування модулів

Клас SubjectMaker має метод:

- request() – робить GET реквест та парсить JSON;

Клас ModuleView має методи:

- init() – ініціацізує вью;
- initWithSubjectModules() – кастомний ініціалізатор, який приймає масив модулів

Клас VideoLayer має методи:

- init() – ініціацізує вью;
- layoutSubviews() – застосовую правила автолайаута;
- setContentAsset () – задає асет для програвання;

- play () – починає програвання;
- pause () – зупиняє програвання;
- previewImageForLocalVideo () – показує 1й кадр відео як картинку;
- rewind () – скролить відео до певного моменту;
- stop () – зупиняє відео;

Клас CustomLogOutCellмає методи:

- awakeFromNib () – завантажує комірку;
- setSelected() – викликає логіку натиску на комірку

Приклад коду класу SubjectViewController():

```
//
//  SubjectViewController.swift
//  RatingSystemTNEU
//
//  Created by dimakomar on 11/3/15.
//  Copyright © 2015 Dima Komar. All rights reserved.
//

import UIKit
import DGRunkeeperSwitch
import PullToMakeFlight
import AVFoundation

class SubjectViewController: UIViewController, UITableViewDataSource, UITableViewDelegate {
    var firstSemesterSubjectArray = [FirstSemestrSubject]()
    var secondSemesterSubjectArray = [FirstSemestrSubject]()
    var dataSource = [FirstSemestrSubject]()

    @IBOutlet weak var myTableView: UITableView!
    var subjects = [String]()
    var date = [SubjectsModules]()
    var realDate = [SubjectsModules]()
    var modulesDates = [String]()

    var modulesCounterArrayOfInt = [Int]()
    var modulesCounter = 0
    var moduleString: String!
    var modulesForSubjects = [[SubjectsModules]]()
    var flightSound:AVAudioPlayer = AVAudioPlayer()

    @IBOutlet weak var woSwitch: DGRunkeeperSwitch!
    @IBOutlet weak var topView: UIView!


    @IBAction func switchChangedValue(sender: DGRunkeeperSwitch) {
```

```
        switch (sender.selectedIndex) {
        case 0:
        myTableView.reloadData()
        self.dataSource.removeAll()
        self.dataSource.appendContentsOf(self.firstSemesterSubjectArray)
        getModulesOutOf(self.dataSource)

        myTableView.reloadData()
        case 1:
        self.dataSource.removeAll()
        self.dataSource.appendContentsOf(self.secondSemesterSubjectArray)
        getModulesOutOf(self.dataSource)
        NSOperationQueue.mainQueue().addOperationWithBlock {
        self.myTableView.reloadData()
        }



        default: self.dataSource = self.secondSemesterSubjectArray
        myTableView.reloadData()

        }
    }

    override func viewDidAppear(animated: Bool) {
        super.viewDidAppear(animated)
    }

    override func viewDidLoad() {
        super.viewDidLoad()


        if self.secondSemesterSubjectArray.count == 0 {
            self.woSwitch.enabled = false
            self.woSwitch.backgroundColor = UIColor.grayColor()
        } else {
            self.woSwitch.backgroundColor = UIColor(red: 0.133, green: 0.561, blue: 0.757, alpha:
1.0)
        }

        print("datasource: \(self.dataSource)")

        self.woSwitch.rightTitle = "2 Семетр"
        self.woSwitch.leftTitle = "1 Семестр"
        self.getModulesOutOf(self.dataSource)



        let flightSoundURL:NSURL = NSBundle.mainBundle().URLForResource("aircraft051",
withExtension: "mp3")!

        myTableView.addPullToRefresh(PullToMakeFlight(), action: { () -> () in

, {[unowned self] in
                self.myTableView.endRefreshing()
                self.flightSound = try! AVAudioPlayer(contentsOfURL: flightSoundURL, fileTypeHint:
nil)
```

```
            self.flightSound.numberOfLoops = 0
            self.flightSound.prepareToPlay()
            self.flightSound.play()
            })
        })

    }


    func getModulesOutOf(semecter: [FirstSemestrSubject]) {
        self.modulesForSubjects.removeAll()


            for moduleInfo in item.modulesArr {

                self.moduleString = (moduleString ?? "") + "\(moduleInfo.date) |"
                self.date.append(moduleInfo)

            }
            self.modulesForSubjects.append(self.date)
            self.modulesCounterArrayOfInt.append(self.date.count)
            self.date = [SubjectsModules]()
            self.moduleString = ""
        }

    }



        if indexPath.row == self.dataSource.count {
            return 80.0
        } else {
            return 160.0
        }
    }

}
```

3.2 Програмна реалізація API

API реалізовано на мові Java . Використовує веб-сервер Tomcat та побудоване за принципом RESTful json api.

Приклад ендпоінта:

/api/getScores?login=*****&password=*****

Приклад відповіді сервера:

```
{
    "student": {
        "name": "Сімашко Олена Володимирівна ",
```

```
        "group": "ПЗАС-21",
        "firstSemesterModule": {
            "subjects": [{
                "studentName": "Вища математика",
                "controlType": "Залік",
                "subjectModules": [{
                    "date": "24.10.14",
                    "weight": 30,
                    "score": 90
                }, {
                    "moduleDate": "22.11.14",
                    "moduleweight": 40,
                    "modulescore": 98
                }, {
                    "moduledate": "26.11.14",
                    "moduleweight": 30,
                    "modulescore": 90
                }]
            }]
        },
        "secondSemesterModule": {
            "modulesubjects": []
        }
    },
    "phpsessid": "rO0ABXNyABdqYXZhLnV0aWwuTGlua2VkSGFza...",
    "success": true
}
```


Класи Module, Semester, Student, Subject є класами, які означають сутність. Вони містяться тільки атрибути. Опис інших класів (ModuleParser, NameGroup, SemesterParser, SubjectsParser) методами, що виконують важливі функції програми, розглянуто нижче.

Клас ModuleParser методи:

- parseModule() – задає дату оцінку і вагу
- parseWeight() – задає вагу
- parseDate() – задає дату

- parseScore() – задає оцінку

Клас NameGroupParserметоди:

- parseName() – задає ім’я студента
- parseGroup() – задає назву групи
- capitalizeString() – робить імена напианими з великої букви

Клас SubjectParser має методи:

- parseSubject() - задає назву, тип контролю, загальну оцінку і модулі
- parseName() – задає назву
- parseControlType() – задає тип контролю
- parseTotalScore() – задає загальну оцінку
- parseModules() – задає модулі

Висновки до розділу 3

1. Обґрунтовано вибір мови програмування, якою являється мова Swift та засобів розробки програмної системи.
2. Організовано інтерфейс користувача із системою.
3. Використано API як джерело данних
4. Описано об’єктно-орієнтований підхід програмування, описавши методи (функції) класів.

РОЗДІЛ 4
ТЕСТУВАННЯ ТА ДОСЛІДНА ЕКСПЛУАТАЦІЯ

4.1 Тестування

Одним із головних етапів є етап тестування. Тестування – це процес оцінки якості розробленого програмного забезпечення .

На даному етапі проведено такі типи тестування:

- функціональне;
- тестування безпеки.

Функціональне тестування відносить до різновиду системного тестування і передбачає перевірку відповідності функціональності системи технічному завданню, тестування продуктивності системи та її стійкості, перевірку роботи системи з вхідними даними.

Провівши функціональне тестування, у таблиці 4.1 представлено результати перевірки всіх варіантів використання, зазначених на етапі аналізу та опису предметної області та реалізації програмної системи.

Таблиця 4.1

Результати проходження функціонально тестування

N/n	Варіант використання	Результат тесту
1.	Авторизація користувача в системі	Пройдено
2.	Зміна семестру	Пройдено
3.	Гортання таблиці	Пройдено
4.	Оновлення Таблиці	Пройдено
5.	Вихід	Пройдено

Продовження таблиці 4.1

6.	Збереження паролю	Пройдено
7.	Відкриття сторінки розробника	Пройдено
8.	Закриття сторінки розробника	Пройдено

У таблиці 4.2 наведено опис тестових випадків (test cases) для основних функцій системи. У ході тестування створено 9 тестових випадків.

Таблиця 4.2

Специфікація тестових випадків

N/n	Тестовий випадок	Результат проходження тесту
1.	Перевірка кнопок	Пройдено
2.	Перевірка прокрутки таблиць	Пройдено
3.	Перевірка заповнення обов’язкових полів	Пройдено
4.	Перевірка максимальної довжини полів	Пройдено
5.	Перевірка виведення повідомлень підтвердження	Пройдено
6.	Перевірка відповідності введених даних у полях	Пройдено
7.	Перевірка веб переглядача	Пройдено
8.	Перевірка анімацій	Пройдено
9.	Перевірка звукових ефектів	Пройдено

Тестування безпеки має велике значення. Добре розроблений захист системи запобігає несанкціонованому доступу до даних.

Програма кешує дані для швидшого доступу тому існує вірогідність доступу до закешованих данних при не правильній логіці відновлення сесії. Було оброблено всі випадки доступу до чужої сесії, тому зараз несанкціонований доступ є неможливим.

Також для тестування безпеки було розроблено 4 тестових випадки (табл. 4.3). Всі випадки пройшли тестування успішно. Програмна система є надійною щодо несанкціонованого доступу до даних.

Таблиця 4.3

Тестові випадки тестування безпеки

N/n	Тестовий випадок	Результат тесту	Тестові дані
1.	Ввід логіну (менше 4 символів)	Пройдено	Vas
2.	Ввід паролю (менше 4 символів)	Пройдено	123
3.	Ввід логіну кирилицею	Пройдено	Вася
4.	Ввід паролю кирилицею	Пройдено	Пароль
5.	Ввід логіну із спеціальним символом	Пройдено	Vasya$$%
6.	Ввід паролю із спеціальним символом	Пройдено	Password$%^
7.	Вхід в систему із вимкненою мережею	Пройдено	-

В тестовому випадку при вводі логіну менше 6 символів з’являється вікно з повідомленням помилки, тому, що логін може мати

не менше 6 символів. Це було передбачено на етапі розробки програмної системи.

В тестовому випадку при вводі паролю менше 6 символів з’являється вікно з повідомленням помилки, тому, що пароль може мати не менше 6 символів. Це було передбачено на етапі розробки програмної системи.

В тестовому випадку при вводі логіну кирилицею з’являється вікно з повідомленням помилки, тому, що логіну може мати лише латинські символи, тире та крапку. Це було передбачено на етапі розробки програмної системи.

В тестовому випадку при вводі паролю кирилицею з’являється вікно з повідомленням помилки, тому, що пароль може мати лише латинські символи, тире та крапку. Це було передбачено на етапі розробки програмної системи.

В тестовому випадку при вводі логіну із спецільним символом з’являється вікно з повідомленням помилки, тому, що логін не може мати спеціальні символи крім тире та крапки. Це було передбачено на етапі розробки програмної системи.

В тестовому випадку при вводі паролю із спецільним символом з’являється вікно з повідомленням помилки, тому, що пароль не може мати спеціальні символи крім тире та крапки. Це було передбачено на етапі розробки програмної системи.

В тестовому випадку при вході в систему із вимкненою мережею з’являється вікно з повідомленням помилки, тому, що для перегляду оцінок потрібний зв’язок з мережою інтернет. Це було передбачено на етапі розробки програмної системи.

На цьому етапі проведено тестування графічного користувацького інтерфейсу. Інтерфейс користувача відповідає прототипам системи, які представлено у підрозділі 1.4.

У таблиці 4.4 приведено тестові випадки для тестування користувацького інтерфейсу.

Таблиця 4.4

Тестові випадки тестування інтерфейсу користувача

N/n	Тестовий випадок	Результат тесту
1.	Перевірка розміщення елементів управління на екранних формах	Пройдено
2.	Перевірка змісту і оформлень повідомлень, що виводяться	Пройдено
3.	Перевірка до форматів введення	Пройдено
4.	Перевірка реакції системи на дії користувача	Пройдено
5.	Перевірка часу відгуку на команди	Пройдено

Тестові випадки, описані у таблиці 4.4, пройдено за допомогою ручного тестування.

Результати проведення тестування представлені у додатку Л.

4.2 Інструкція користувача

Компоненти ПЗ

Пакет розроблений на мові програмування Swift у середовищі розробки Xcode 7.3 і може експлуатуватись під управління

операційною системою iOS. Під час проектування та розробки програми було використано об’єктно-орієнтований підхід.

Для коректної роботи пакету необхідний iPhone 4s+ із налаштованим Apple ID.

Встановлення ПЗ

ПЗ розповсюджується через App Store (Магазин додатків Apple)

Базові функції ПЗ

Для входу в систему потрібно ввести логін та пароль (рис.4.1). Після введення паролю можна вибрати чи хочете ви зберігати пароль кожного разу. Після введення цих даних відкривається вікно для створення та перегляду оцінок.

(рис.4.2).

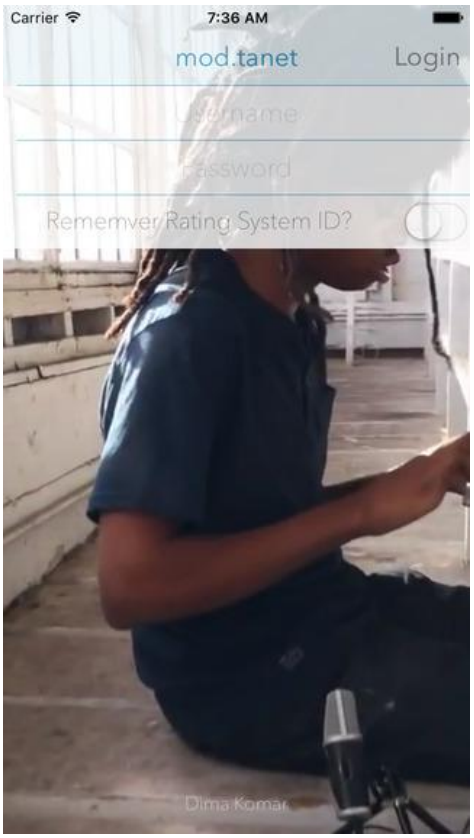


Рисунок 4.1 – Вікно авторизації

На рисунку 4.2 зображено вікно перегляду оцінок. Зверху зображений перемикач семестрів. Нижче таблиця предметів. Кожен предмет має кількість модулів (яка може відрізнятись, залежно від предмету). Кожен предмет має загальний бал який зображено з правої частини екрану

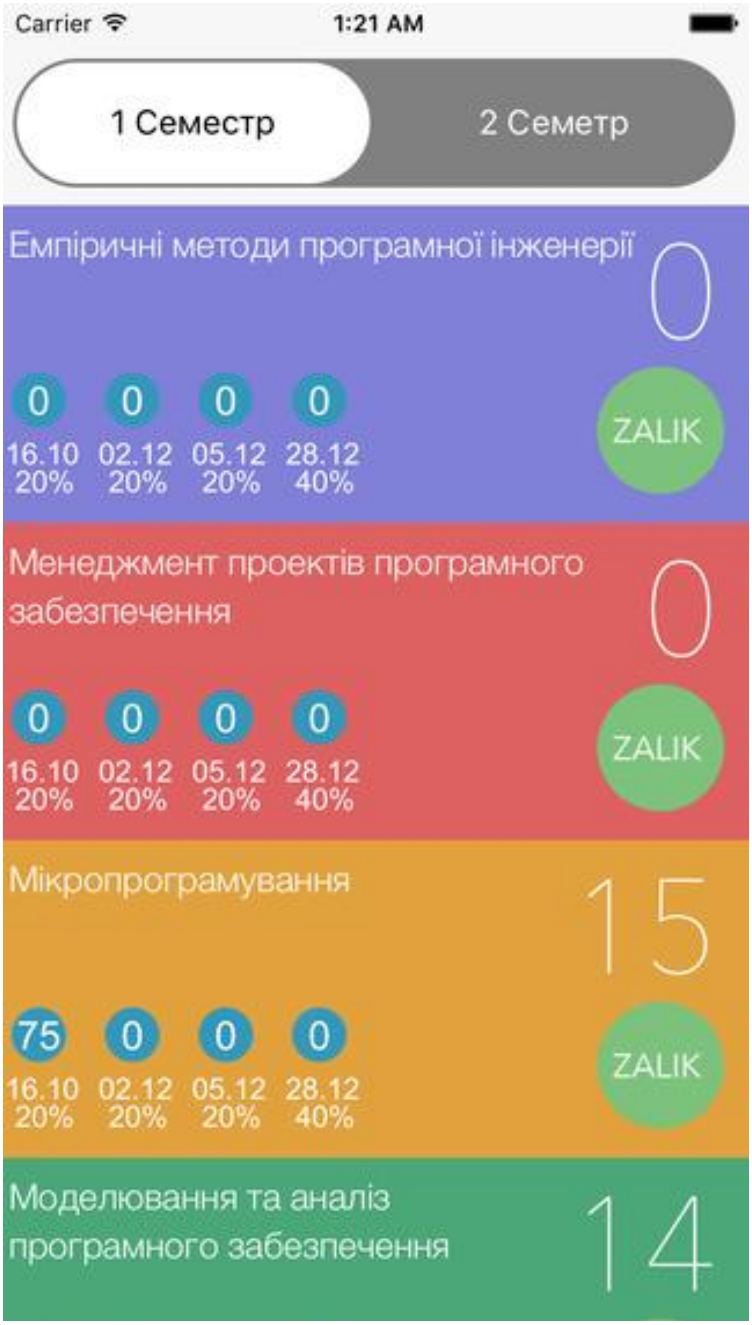


Рисунок 4.2 – Вікно додавання салону

Потягнувши слайдер зверху можно змінити семестр (рис.4.3).

Рисунок 4.3 – Головний екран перегляду предметів

Потягнувши таблицю вниз можна побачити слайдер оновлення у вигляді літака (рис.4.4).

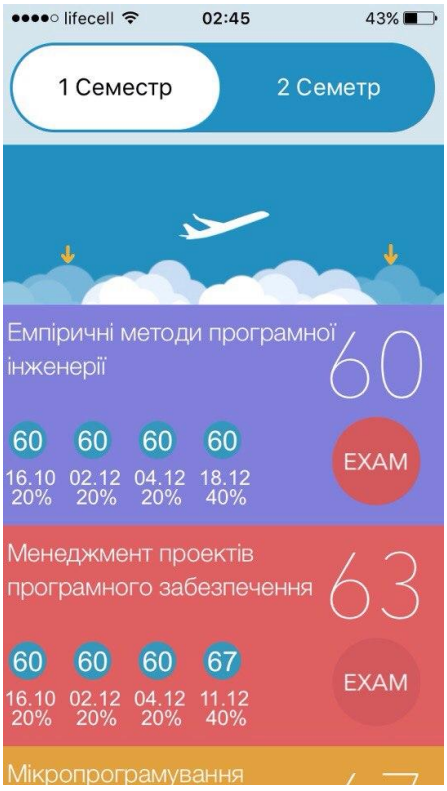


Рисунок 4.4 – Панель інструментів опції «Додати»

Потягнувши таблицю вверх до кінця можна побачити кнопку виходу із профайлу (рис.4.5).

Рисунок 4.5 – Вікно додавання нового відділу

Висновки до розділу 4

1. Проведено модульне, функціональне та GUI тестування програмної системи.
2. Описано інструкцію користувача програмної системи.

ВИСНОВКИ

Під час створення програмної системи у дипломній роботі було пройдено усі етапи, а саме:
На етапі аналізу предметної області було досліджено предметну область – система оцінок для студентів ТНЕУ. Проаналізовано та охарактеризовано систему оцінок, оглянуто аналіз існуючих аналогів програмної системи, визначено функціональні та нефункціональні вимоги.

На етапі проектування програмної системи розроблено її архітектуру та представлено у вигляді класів програми. Вивчено інтерфейс серверної частини та способи інтеграції.

На етапі програмної реалізації, з використанням об’єктно-орієнтованого підходу в інтегрованому середовищі розробки програмного забезпечення Xcode 7.3, реалізовано функції системи, які написані на мові програмування Swift з імплементацією запитів до серверу. Розроблено інтерфейс програми за допомогою Storyboard.

На етапі тестування було проведено GUI, функціональне та тестування безпеки. Створені тест-випадки для кожного виду тестування. Також розроблено інструкцію користувача, де вказується правила використання програми.

Програмна система дає можливість швидкого та зручного доступу до системи оцінок, що значно покращує та полегшує навчання кожного студента. Після затвердження її керівництвом університету, дану програму буде введено в подальшу експлуатацію.

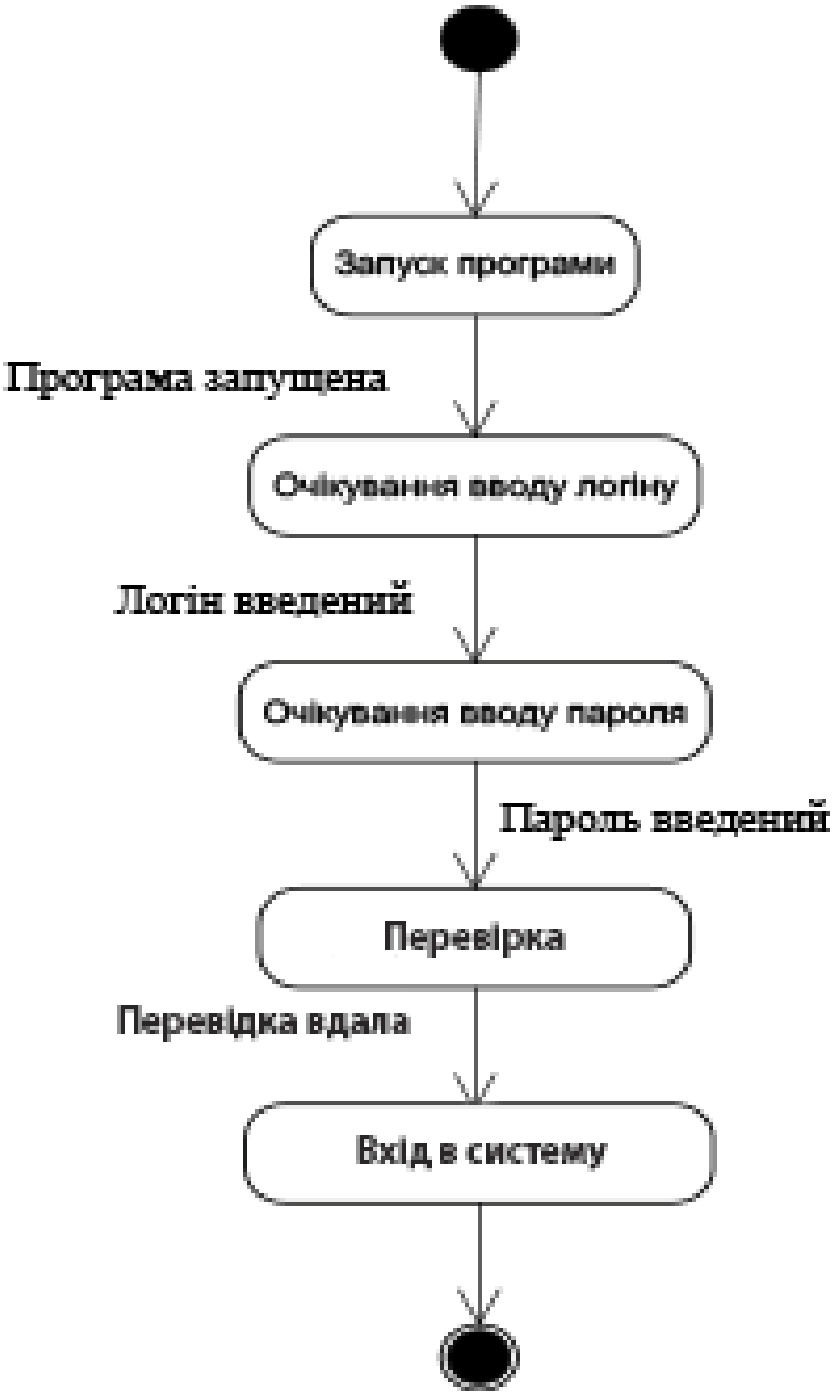
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. The Swift Programming Language (Swift 2.2) : Apple Inc, 2014. – 500 с.
2. Using Swift with Cocoa and Objective-C (Swift 2.2): Apple Inc, 2014. – 500 с.
3. iOS Human Interface Guidelines : Apple Inc, 2014. – 500 с.
4. Процес [Електронний ресурс]. – Режим доступу: http://library.if.ua/book/28/1897.html
5. Логін і пароль [Електронний ресурс]. – Режим доступу: http://yak-prosto.com/yak-pridumati-login-dlya-poshti/
6. Контекстна діаграма потоків даних [Електронний ресурс]. – Режим доступу: http://ukrdoc.com.ua/text/1632/index-1.html?page=7
7. Діаграма сутностей-зв’язків [Електронний ресурс]. – Режим доступу: http://bseu.by/it/tohod/lekcii4_3.htm
8. Діаграма станів [Електронні ресурси]. – Режим доступу: http://bug.kpi.ua/stud/work/RGR/UML/stanu&actyv.html
9. Информационные системы и технологии в экономике и управлении: учебник/ под ред. Проф.В. В. Трофимова. - 2-е изд., перераб. и доп. - М.: Высшее образование, 2007. – 207с.
10. Аглицкий Д.С., Аглицкий И.С. Рынок информационных технологий: проблемы и решения. - М.:2000. – 168 с.
11. Бізнес-процес [Електронний ресурс]. – Режим доступу: http://www.rusnauka.com/6_PNI_2012/Economics/6_102471.doc.htm

12. Cocoa and Cocoa Touch are the application development: https://developer.apple.com/library/ios/documentation/General/Conceptual/DevPedia-CocoaCore/Cocoa.html
13. Основи та переваги мови swift apple [Електронний ресурс]. – Режим доступу: http://idev-swift.ru/documentation-on-the-russian-swift/основы-и-преимущества-языка-swift-apple/
14. Тестування програмного забезпечення [Електронний ресурс]. – Режим доступу: http://lib.mdpu.org.ua/e-book/vstup/L11.htm
15. Функціональне тестування [Електронний ресурс]. – Режим доступу: http://www.programsfactory.univ.kiev.ua/content/books/2/91

ДОДАТОК А

Схема бізнес процесу



Бізнес процес «Авторизація»

ДОДАТОК Б

Специфікація вимог для програмної системи iOS додатку для перегляду оцінок студентів ТНЕУ+

1. Вступ

1.1 Призначення і мета

Програмна система призначення для обліку та контролю оцінок студентів. Основним завданням даної системи є полегшення та спрощення доступу до системи оцінок через зручний та інтуїтивно зрозумілий інтерфейс.

1.2 Продукти-алоги

Порівняльна характеристика продуктів зображена у таблиці 1.

Таблиця 1

Порівняльна характеристика програмних продуктів

Назва програмного продукту	ModuleOK	Аналитика Оценок
Фірма-розробник	Україна, Тернопіль	Росія, Москва
Функціональність	Програмний продукт дозволяє: 6) Перегляд оцінок; 7) Перегляд графіку успішності	Програмний продукт вміщує: 2) Перегляд оцінок;

Продовження Таблиці 1

Інтерфейс користувача	Інтерфейс даної програми не зовсім зручний. Мова програми – українська. Система недоступна для iOS та OS X	Інтерфейс програми є зручним у використанні. У головному вікні програми висвітлені всі пункти меню роботи. Система доступна тільки для Росії
Допомога користувачу	В даному програмному продукті розробник не надав жодної допомоги користувачеві.	В головного меню програми є пункт «Помощь», де описано інформацію про програму, версію та контактні дані для додаткових питань стосовно використання. Також в додатку до програми входить презентація покрокової роботи з програмою.

2. Загальний опис

2.1 Характеристика продукту

Згідно із варіантами використання, до складу програмної системи перегляду оцінок входить наступний перелік функцій:

- авторизація користувачів у систему;
- збереження логіну па паролю;
- перегляд оцінок;
- перегляд ваги оцінок;
- перегляд дат модулів;
- перегляд загальної оцінки;
- перегляд назви дисципліни;
- перемикання семестрів;
- вихід із системи;

2.2 Класи користувачів

Користувачам програмна система надає доступ управління даними на рівні Студент:

Студент – має право на перегляд оцінок, збереження свого логіну та паролю, входу\виходу в систему.

2.3 Середовище існування

Дана програма буде функціонувати на базі операційної системи iOS 8 та вище та на девайсах iPhone 4s та вище або iPad 2 та вище.

3. Характеристики системи

REQ-3.1 Характеристики вимоги «Авторизація»

3.1.1 Опис і пріоритет:

Рівень пріоритету «Обов’язкова» та низька складність реалізації.

3.1.2 Послідовність дій:

1) Запуск програми.
2) Введення логіну та паролю.

3.1.3 Функціональні вимоги:

1) Користувач зареєстрований в систему;

REQ-3.2 Характеристики вимоги «Зміна Семестру»

3.2.1 Опис і пріоритет:

Рівень пріоритету «Обов’язковий» та середня складність реалізації.

3.2.2 Послідовність дій:

1) Запуск програми;
2) Введення логіну та паролю;
3) Авторизація

3.2.3 Функціональні вимоги:

1) Вдала авторизація;

REQ-3.3 Характеристики вимоги «Перегляд оцінок для обох семестрів»

3.3.1 Опис і пріоритет:

Рівень пріоритету «Обов’язковий» та низька складність реалізації.

3.3.2 Послідовність дій:

1) Запуск програми;
2) Введення логіну та паролю;
3) Авторизація;

3.3.3 Функціональні вимоги:

1) Вдала авторизація;

REQ-4 Характеристика вимоги «Вихід»

3.4.1 Опис і пріоритет:

Рівень пріоритету «Обов’язковий» та низька складність реалізації.

3.4.2 Послідовність дій:

1) Запуск програми;
2) Введення логіну та паролю;
3) Авторизація.

3.4.3 Функціональні вимоги:

1) Вдала авторизація

REQ-5 Характеристика вимоги «Збереження логіну та паролю»

3.5.1 Опис і пріоритет:

Рівень пріоритету «Обов’язковий» та низька складність реалізації.

3.5.2 Послідовність дій:

1) Запуск програми;
2) Введення логіну та паролю;
3) Авторизація.

3.5.3 Функціональні вимоги:

1) Вхід у додаток
2) Вдала авторизація;

4. Інші нефункціональні вимоги

4.1 Вимоги продуктивності

Операційна система – iOS

Девайс – iPhone 4s або iPad 2 та вище

Вільна пам’ять: більше 50 мегабайт

4.2 Вимоги безпеки

З метою забезпечення безпеки та цілісності збереження даних, користувачам рекомендується використовувати паролі, які складають

не менше 10 символів. Також для безпеки рекомендується міняти паролі через кожні 42 дні.

4.3 Атрибути якості програмного продукту

Програмна система для обліку та перегляду оцінок для студентів ТНЕУ забезпечити легкість адаптації до змін в середовищі операційної системи. Система повинна бути стійкою до збоїв та забезпечувати збереження даних при них.

ДОДАТОК В

Глосарій

Термін	Опис терміну
Основні поняття та категорії предметної області та проекту	
Програмне забезпечення	Сукупність програм системи обробки інформації і програмних документів, необхідних для експлуатації цих програм.
Програмний продукт	Програма, яку може запускати, тестувати, виправляти та змінювати будь-яка людина. Вона може використовуватись в різних операційних системах та з різними наборами даних.
Бізнес-процес	Будь-яка діяльність, що має вхідний продукт, додає вартість до нього, та забезпечує вихідний продукт для внутрішнього або зовнішньогоспоживача.[3]
Обробка даних	Систематична цілеспрямована послідовність дій над даними.
Користувачі системи	
Студент	Особа яка навчається в університеті
Вхідні та вихідні документи	

Продовження Таблиці “Голосарій”

Логін	Алфавітно-цифровий набір символів, що ідентифікує користувача комп'ютерної мережі і разом із паролем використовується операційною системою для надання йому дозволу на з'єднання з комп'ютерною системою та визначення його прав доступу до ресурсів мережі. Логін має бути унікальним в межах даної системи.[6]
Пароль	Секретне слово або певна послідовність символів, призначена для підтвердження особи або її прав. Паролі використовують для захисту інформації від несанкціонованого доступу. Паролі разом із логіном використовуються операційною системою для надання користувачу дозволу на з'єднання з комп'ютерною системою та визначення його прав доступу до ресурсів мережі.
Маніпуляція даними	Здійснення певних функцій над даними (додавання, редагування, видалення. пошук), певна обробка даних.

Продовження Таблиці “Голосарій”

API	Прикладни́й програ́мний інтерфе́йс (інтерфейс програмування додатків, інтерфейс прикладного програмування) (англ. Application Programming Interface, API) — набір визначень взаємодії різнотипного програмного забезпечення. API — це зазвичай (але не обов'язково) метод абстракції між низькорівневим та високорівневим програмним забезпеченням.

ДОДАТОК Д

Діаграма варіантів використання

ДОДАТОК Е

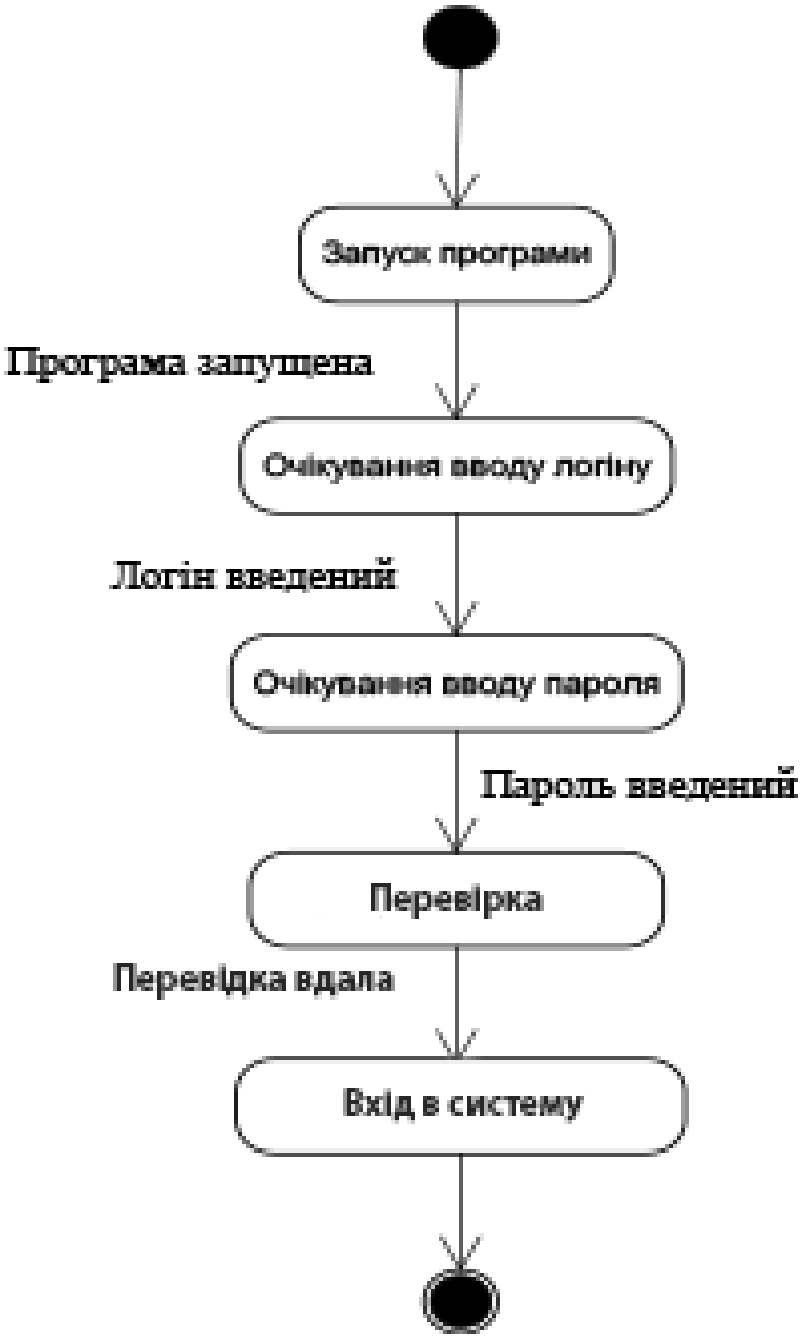
Варіанти використання

Варіант використання «Перегляд оцінок»

Контекст використання	Авторизація та перегляд оцінок.
Дійові особи	Студент.
Передумова	Відсутня.
Триггер	Відкрите вікно авторизації користувача.
Сценарій	Заповнення всіх полів вікна авторизації; Натиснення кнопки «Ввійти»;
Постумова	При правильно заповнених полях буде здійснено вхід в систему. При неправильному введені даних вхід не відбудеться, необхідно заново заповнити потрібні поля.

ДОДАТОК Ж

Діаграми станів



Діаграма станів авторизації користувача в систему

ДОДАТОК К

Лістинг програми

Лістинг коду класу SubjectViewController.swift

```
//
//  SubjectViewController.swift
//  RatingSystemTNEU
//
//  Created by dimakomar on 11/3/15.
//  Copyright © 2015 Dima Komar. All rights reserved.
//

import UIKit
import DGRunkeeperSwitch
import PullToMakeFlight
import AVFoundation

class    SubjectViewController:    UIViewController,    UITableViewDataSource,
UITableViewDelegate {
    var firstSemesterSubjectArray = [FirstSemestrSubject]()
    var secondSemesterSubjectArray = [FirstSemestrSubject]()
    var dataSource = [FirstSemestrSubject]()

    @IBOutlet weak var myTableView: UITableView!
    var subjects = [String]()
    var date = [SubjectsModules]()
    var realDate = [SubjectsModules]()
    var modulesDates = [String]()

    var modulesCounterArrayOfInt = [Int]()
    var modulesCounter = 0
    var moduleString: String!
    var modulesForSubjects = [[SubjectsModules]]()
    var flightSound:AVAudioPlayer = AVAudioPlayer()
```

```
@IBOutlet weak var woSwitch: DGRunkeeperSwitch!
@IBOutlet weak var topView: UIView!


@IBAction func switchChangedValue(sender: DGRunkeeperSwitch) {

    switch (sender.selectedIndex) {
    case 0:
    myTableView.reloadData()
    self.dataSource.removeAll()
    self.dataSource.appendContentsOf(self.firstSemesterSubjectArray)
    getModulesOutOf(self.dataSource)

    myTableView.reloadData()
    case 1:
    self.dataSource.removeAll()
    self.dataSource.appendContentsOf(self.secondSemesterSubjectArray)
    getModulesOutOf(self.dataSource)
    NSOperationQueue.mainQueue().addOperationWithBlock {
    self.myTableView.reloadData()
    }



    default: self.dataSource = self.secondSemesterSubjectArray
    myTableView.reloadData()

    }
}

override func viewDidAppear(animated: Bool) {
    super.viewDidAppear(animated)
}

override func viewDidLoad() {
    super.viewDidLoad()
```

```
        if self.secondSemesterSubjectArray.count == 0 {
            self.woSwitch.enabled = false
            self.woSwitch.backgroundColor = UIColor.grayColor()
        } else {



        self.myTableView.backgroundColor = UIColor(red: 0.133, green: 0.561,
blue: 0.757, alpha: 1.0)
        myTableView.registerNib(UINib(nibName: "CustomOneCell", bundle: nil),
forCellReuseIdentifier: "CustomCellOne")
        myTableView.registerNib(UINib(nibName:   "CustomLogOutCell",   bundle:
nil), forCellReuseIdentifier: "CustomLogOutCell")
        myTableView.separatorStyle = UITableViewCellSeparatorStyle.None
        self.topView.backgroundColor = UIColor(red: 0.961, green: 0.961, blue:
0.961, alpha: 1.0)
        self.myTableView.contentInset = UIEdgeInsetsMake(101, 0, 0, 0)
        myTableView.layoutIfNeeded()

        let                        flightSoundURL:NSURL                        =
NSBundle.mainBundle().URLForResource("aircraft051", withExtension: "mp3")!

        myTableView.addPullToRefresh(PullToMakeFlight(), action: { () -> () in
            let delayTime = dispatch_time(DISPATCH_TIME_NOW,
                Int64(1 * Double(NSEC_PER_SEC)))
            dispatch_after(delayTime,   dispatch_get_main_queue(),   {[unowned
self] in
                self.myTableView.endRefreshing()
                self.flightSound    =    try!    AVAudioPlayer(contentsOfURL:
flightSoundURL, fileTypeHint: nil)
                self.flightSound.numberOfLoops = 0
                self.flightSound.prepareToPlay()
                self.flightSound.play()
                })
        })

    }
```

```
    func getModulesOutOf(semecter: [FirstSemestrSubject]) {
        self.modulesForSubjects.removeAll()
        for item in self.dataSource {

            for moduleInfo in item.modulesArr {

                self.moduleString = (moduleString ?? "") + "\(moduleInfo.date)
|"
                self.date.append(moduleInfo)

            }
            self.modulesForSubjects.append(self.date)
            self.modulesCounterArrayOfInt.append(self.date.count)
            self.date = [SubjectsModules]()
            self.moduleString = ""
        }

    }


     func tableView(tableView: UITableView, numberOfRowsInSection section:
Int) -> Int {
        return self.dataSource.count + 1
    }
    func tableView(tableView: UITableView, didSelectRowAtIndexPath indexPath:
NSIndexPath) {
         dispatch_async(dispatch_get_main_queue()) {
        self.performSegueWithIdentifier("LogOut", sender: self)
        }
    }

     func tableView(tableView: UITableView, cellForRowAtIndexPath indexPath:
NSIndexPath) -> UITableViewCell {

        if indexPath.row == self.dataSource.count  {
            let                              cell                              =
tableView.dequeueReusableCellWithIdentifier("CustomLogOutCell",  forIndexPath:
```

```
indexPath) as! CustomLogOutCell
            return cell
        } else {

        let    cell    =    tableView.dequeueReusableCellWithIdentifier("Cell",
forIndexPath: indexPath)

        let         woView         =         ModuleView(initWithSubjectModules:
modulesForSubjects[indexPath.row],                              actualDataSource:
self.dataSource[indexPath.row],   andFrame:   CGRect(x:   0   ,   y:   0,   width:
cell.frame.width, height: cell.frame.height))
        // remove subviews

        for subview in cell.subviews{
            subview.removeFromSuperview()

        }

        var arrayOfColors = [UIColor(red: 0.5, green: 0.5, blue: 0.85, alpha:
1.0), UIColor(red: 0.87, green: 0.38, blue: 0.38, alpha: 1.0), UIColor(red:
0.882, green: 0.635, blue: 0.243, alpha: 1.0), UIColor(red: 0.294, green:
0.655, blue: 0.471, alpha: 1.0), UIColor(red: 0.706, green: 0.447, blue:
0.357, alpha: 1.0), UIColor(red: 0.5, green: 0.5, blue: 0.85, alpha: 1.0),
UIColor(red: 0.3, green: 0.655, blue: 0.471, alpha: 1.0), UIColor(red: 0.882,
green: 0.635, blue: 0.243, alpha: 1.0), UIColor(red: 0.294, green: 0.655,
blue: 0.471, alpha: 1.0), UIColor(red: 0.706, green: 0.447, blue: 0.357,
alpha: 1.0)]

        //cell.backgroundColor = UIColor.orangeColor()
        // then add your view
        cell.backgroundColor = arrayOfColors[indexPath.row]
        cell.addSubview(woView)
        return cell

        }
    }

    func     tableView(tableView:     UITableView,     willDisplayCell     cell:
```

```
UITableViewCell, forRowAtIndexPath indexPath: NSIndexPath) {


        let rotationTransform = CATransform3DTranslate(CATransform3DIdentity,
-30, 0, 0)
        cell.layer.transform = rotationTransform
        UIView.animateWithDuration(0.4, animations: { cell.layer.transform =
CATransform3DIdentity })

    }

     func tableView(tableView: UITableView, shouldHighlightRowAtIndexPath
indexPath: NSIndexPath) -> Bool {
        if indexPath.row == self.dataSource.count {
            return true
        } else {
            return false
        }
    }

     func tableView(tableView: UITableView, heightForRowAtIndexPath indexPath:
NSIndexPath) -> CGFloat {
        if indexPath.row == self.dataSource.count {
            return 80.0
        } else {
            return 160.0
        }
    }

}
```

Лістинг коду класу LoginViewController.swift

```
//
//  LoginController.swift
//  RatingSystemTNEU
//
```

```
//  Created by dimakomar on 11/3/15.
//  Copyright © 2015 Dima Komar. All rights reserved.
//

import UIKit
import QuartzCore
import SVProgressHUD
import ReachabilitySwift


let useClosures = false
class LoginController: UITableViewController {

    //MARK: Outlets for UI Elements.
    @IBOutlet weak var usernameField:   UITextField!
    @IBOutlet weak var passwordField:   UITextField!
    @IBOutlet weak var mySwitch: UISwitch!
    @IBOutlet weak var loginButton: UIButton!

    var reachability: Reachability?
    var firstSemesterSubjectArray = [FirstSemestrSubject]()
    var secondSemesterSubjectArray = [FirstSemestrSubject]()
    var decodedFirstSemesterSubjectArray = [FirstSemestrSubject]()
    var decodedSecondSemesterSubjectArray = [FirstSemestrSubject]()
    var subjects = [String]()
    //MARK: Global Variables for Changing Image Functionality.
    private var idx: Int = 0

    var secondRunLoop = false
    var someKingOfError = false
    let userDefaults = NSUserDefaults.standardUserDefaults()

    //MARK: View Controller LifeCycle



    override func viewWillAppear(animated: Bool) {
        super.viewWillAppear(true)
```

```
        setupReachability()

        SVProgressHUD.setBackgroundColor(UIColor(red: 0.1, green: 0.1, blue:
0.2, alpha: 0.6))
        SVProgressHUD.setForegroundColor(UIColor(red: 0.9, green: 0.9, blue:
0.9, alpha: 1))

        usernameField.addTarget(self,       action:       "textFieldDidChange",
forControlEvents: UIControlEvents.EditingChanged)
        passwordField.addTarget(self,       action:       "textFieldDidChange",
forControlEvents: UIControlEvents.EditingChanged)

        // Visual Effect View for background
        let visualEffectView = UIVisualEffectView(effect: UIBlurEffect(style:
UIBlurEffectStyle.Dark)) as UIVisualEffectView
        visualEffectView.frame = self.view.frame
        visualEffectView.alpha = 0.5


    }

    override func viewDidLoad() {
        super.viewDidLoad()
        loginButton.setTitleColor(UIColor.grayColor(),               forState:
UIControlState.Disabled)

        for view in tableView.subviews {
            if view is UIScrollView {
                (view as? UIScrollView)!.delaysContentTouches = false
                break
            }
        }
    }

    override func viewDidAppear(animated: Bool) {
        super.viewDidAppear(true)
```

```
        let                         myOutputUsername                        =
NSUserDefaults.standardUserDefaults().objectForKey("Username")
        let                         myOutputPassword                        =
NSUserDefaults.standardUserDefaults().objectForKey("Password")

        if (myOutputUsername != nil && myOutputPassword != nil)
        {

            self.usernameField.text = myOutputUsername as? String
            self.passwordField.text = myOutputPassword as? String
            self.mySwitch.on =  true
        }

        if usernameField.text!.isEmpty || passwordField.text!.isEmpty

        {
            self.loginButton(false)
        }
        else
        {
            self.loginButton(true)
        }

    }

    func textFieldDidChange() {



        if usernameField.text!.isEmpty || passwordField.text!.isEmpty

        {
             self.loginButton(false)
        }
        else
        {
            self.loginButton(true)
        }
```

```
    }


    func loginButton(enabled: Bool) -> () {
        func enable(){

            UIView.animateWithDuration(1,       delay:       0,       options:
UIViewAnimationOptions.CurveEaseIn, animations: {
                self.loginButton.enabled = true
                }, completion: nil)



        }
        func disable(){

            UIView.animateWithDuration(1,       delay:       0,       options:
UIViewAnimationOptions.CurveEaseOut, animations: {
                self.loginButton.enabled = false
                }, completion: nil)
        }
        return enabled ? enable() : disable()
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
    }

    @IBAction func buttonPressed(sender: AnyObject) {
        let                              stringAll                              =
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLKMNOPQRSTUVWXYZ0123456789_."
        var isUsernameEng = true
        var isPasswrodEng = true

        for  char : Character in (self.usernameField.text?.characters)! {

            if !stringAll.containsString(String(char)) {
                isUsernameEng = false
```

```
            }

        }

        for  char : Character in (self.passwordField.text?.characters)! {

            if !stringAll.containsString(String(char)) {
                isPasswrodEng = false
            }

        }

        if mySwitch.on  {
            let username = self.usernameField.text
            let password = self.passwordField.text
            NSUserDefaults.standardUserDefaults().setObject(username,
forKey:"Username")
            NSUserDefaults.standardUserDefaults().setObject(password,
forKey:"Password")
            NSUserDefaults.standardUserDefaults().synchronize()
        } else {

NSUserDefaults.standardUserDefaults().removeObjectForKey("Username")

NSUserDefaults.standardUserDefaults().removeObjectForKey("Password")
            NSUserDefaults.standardUserDefaults().synchronize()
        }

            if isUsernameEng && isPasswrodEng {
                if self.reachability!.isReachable() {
                SVProgressHUD.show()
                let instance = SubjectMaker()

                instance.request(self.usernameField.text!,
password:self.passwordField.text!)        {        firstSemesterSubjectArray,
secondSemesterSubjectArray, success, server in

                    if server == false {
```

```
                        print("refreshed")


                        if         let         decodedFirst                  =
self.userDefaults.objectForKey("first") as? NSData {
                            self.decodedFirstSemesterSubjectArray            =
NSKeyedUnarchiver.unarchiveObjectWithData(decodedFirst)                     as!
[FirstSemestrSubject]
                        }
                        if         let         decodedSecond                 =
self.userDefaults.objectForKey("second") as? NSData {
                            self.decodedSecondSemesterSubjectArray           =
NSKeyedUnarchiver.unarchiveObjectWithData(decodedSecond)                    as!
[FirstSemestrSubject]
                        }

                        self.someKingOfError = true
                        let   alertController1   =   UIAlertController(title:
"Технічні роботи на mod.tanet", message: "Please try again in 5 seconds",
preferredStyle: .Alert)

                        // Create the actions
                        let  okAction  =  UIAlertAction(title:  "OK",  style:
UIAlertActionStyle.Default) {
                            UIAlertAction in
                            NSLog("OK Pressed")
                        }


                        alertController1.addAction(okAction)
                        NSOperationQueue.mainQueue().addOperationWithBlock {
                            SVProgressHUD.dismiss()
                            if self.decodedFirstSemesterSubjectArray.count > 0
{

                                self.performSegueWithIdentifier("Login",
sender: self)
```

```
                            } else {
                            self.secondRunLoop = true
                            self.performSelector(Selector("buttonPressed:"),
withObject: self, afterDelay: 0.1)

                            }
                        }
                    }

                    if success    {
                        NSOperationQueue.mainQueue().addOperationWithBlock {

                            if firstSemesterSubjectArray.count > 0 {
                            let                     first                     =
NSKeyedArchiver.archivedDataWithRootObject(firstSemesterSubjectArray)
                            self.userDefaults.setObject(first,         forKey:
"first")
                            let                     second                    =
NSKeyedArchiver.archivedDataWithRootObject(secondSemesterSubjectArray)
                            self.userDefaults.setObject(second,        forKey:
"second")
                            self.userDefaults.synchronize()

                            if        let        decodedFirst                =
self.userDefaults.objectForKey("first") as? NSData {
                                self.decodedFirstSemesterSubjectArray        =
NSKeyedUnarchiver.unarchiveObjectWithData(decodedFirst)                    as!
[FirstSemestrSubject]
                            }
                            if        let        decodedSecond               =
self.userDefaults.objectForKey("second") as? NSData {
                                self.decodedSecondSemesterSubjectArray       =
NSKeyedUnarchiver.unarchiveObjectWithData(decodedSecond)                   as!
[FirstSemestrSubject]
                            }

                            self.firstSemesterSubjectArray                   =
firstSemesterSubjectArray
```

```
                            self.secondSemesterSubjectArray                    =
secondSemesterSubjectArray
                            } else {
                                if self.decodedFirstSemesterSubjectArray.count
> 0 {
                                    self.firstSemesterSubjectArray           =
self.decodedFirstSemesterSubjectArray
                                    self.secondSemesterSubjectArray          =
self.decodedSecondSemesterSubjectArray
                                } else {
                                    let           alertController1           =
UIAlertController(title: "Технічні роботи на mod.tanet", message: "Please try
again in 5 seconds", preferredStyle: .Alert)
                                    let okAction = UIAlertAction(title: "OK",
style: UIAlertActionStyle.Default) {
                                        UIAlertAction in
                                        NSLog("OK Pressed")
                                    }
                                    alertController1.addAction(okAction)

self.presentViewController(alertController1, animated: true, completion: nil)
                                }

                            }
                            if self.decodedFirstSemesterSubjectArray.count > 0
{

                            self.performSegueWithIdentifier("Login",   sender:
self)
                            SVProgressHUD.dismiss()
                            } else {
                                let             alertController1             =
UIAlertController(title: "Технічні роботи на mod.tanet", message: "Please try
again in 5 seconds", preferredStyle: .Alert)
                                let  okAction  =  UIAlertAction(title:  "OK",
style: UIAlertActionStyle.Default) {
                                    UIAlertAction in
                                    NSLog("OK Pressed")
```

```
                                }
                                alertController1.addAction(okAction)

                                self.presentViewController(alertController1,
animated: true, completion: nil)
                            }
                        }
                    } else if !self.secondRunLoop {

                        let alertController1 = UIAlertController(title:
"Неправильний логін або пароль", message: "Incorrect login and password",
preferredStyle: .Alert)

                        // Create the actions
                        let okAction = UIAlertAction(title: "OK", style:
UIAlertActionStyle.Default) {
                            UIAlertAction in
                            NSLog("OK Pressed")
                        }

                        alertController1.addAction(okAction)
                         NSOperationQueue.mainQueue().addOperationWithBlock {
                        SVProgressHUD.dismiss()
                            if !self.secondRunLoop {
                        self.presentViewController(alertController1, animated:
true, completion: nil)
                            }
                            }
                        }

                }
                } else {
                    SVProgressHUD.dismiss()
                    let alertController = UIAlertController(title: "Please
connect network", message: "Incorrect login and password", preferredStyle:
.Alert)

                    // Create the actions
```

```
                    let okAction = UIAlertAction(title: "OK", style:
UIAlertActionStyle.Default) {
                        UIAlertAction in
                        NSLog("OK Pressed")
                    }

                    alertController.addAction(okAction)

                    self.presentViewController(alertController,      animated:
true, completion: nil)
                }

            } else {
                SVProgressHUD.dismiss()
                let alertController = UIAlertController(title: "Недопустимий
символ в логіні чи паролі", message: "Incorrect login or password",
preferredStyle: .Alert)

                // Create the actions
                let okAction = UIAlertAction(title: "OK", style:
UIAlertActionStyle.Default) {
                    UIAlertAction in
                    NSLog("OK Pressed")
                }

                alertController.addAction(okAction)

                self.presentViewController(alertController, animated: true,
completion: nil)
            }

        }

    deinit {

        reachability?.stopNotifier()
        if (!useClosures) {
            NSNotificationCenter.defaultCenter().removeObserver(self, name:
```

```
ReachabilityChangedNotification, object: nil)
        }
    }

    override func touchesBegan(touches: Set<UITouch>, withEvent event:
UIEvent?) {

        let touch = touches.first

        if touch?.phase == UITouchPhase.Began {
          self.usernameField.resignFirstResponder()
          self.passwordField.resignFirstResponder()
        }

    }


    override func prepareForSegue(segue: UIStoryboardSegue, sender:
AnyObject?) {
        let destView: SubjectViewController = segue.destinationViewController
as! SubjectViewController


            destView.firstSemesterSubjectArray                                =
self.decodedFirstSemesterSubjectArray
            destView.secondSemesterSubjectArray                               =
self.decodedSecondSemesterSubjectArray
            destView.dataSource = self.decodedFirstSemesterSubjectArray


    }

    @IBAction func backgroundPressed(sender: AnyObject) {
        usernameField.resignFirstResponder()
        passwordField.resignFirstResponder()

    }
```

```
     func setupReachability() {
        do {
            self.reachability                        =                        try
Reachability.reachabilityForInternetConnection()
        } catch {
            print("Cannot setup reachability monitoring")
            return
        }

        self.reachability!.whenReachable = { reachability in

        }
        self.reachability!.whenUnreachable = { reachability in

        }


dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT,  0))
{
        do {

            try self.reachability!.startNotifier() } catch {
            print("Cannot start reachability monitoring")
            return
        }
        }
        print("Started reachability")
    }

    override          func          tableView(tableView:          UITableView,
shouldHighlightRowAtIndexPath indexPath: NSIndexPath) -> Bool {
        return false
    }

}

//Extension for Color to take Hex Values
```

```
extension UIColor{

    class func colorWithHex(hex: String, alpha: CGFloat = 1.0) -> UIColor {
        var rgb: CUnsignedInt = 0;
        let scanner = NSScanner(string: hex)

        if hex.hasPrefix("#") {
            // skip '#' character
            scanner.scanLocation = 1
        }
        scanner.scanHexInt(&rgb)

        let r = CGFloat((rgb & 0xFF0000) >> 16) / 255.0
        let g = CGFloat((rgb & 0xFF00) >> 8) / 255.0
        let b = CGFloat(rgb & 0xFF) / 255.0

        return UIColor(red: r, green: g, blue: b, alpha: alpha)
    }
}
```

ДОДАТОК М

Лістинг API-коду серверу

Лістинг коду класу ModuleParser.java

```
package main.java.api.parser;

import main.java.api.parser.pojo.Module;
import org.jsoup.nodes.Element;

public class ModuleParser {

    private Element moduleElement;

    public ModuleParser(Element moduleElement) {
        this.moduleElement = moduleElement;
    }

    public Module parseModule() {
        Module module = new Module();
        module.setDate(parseDate());
        module.setScore(parseScore());
        module.setWeight(parseWeight());
        return module;
    }

    private int parseWeight() {
        return Integer.parseInt(moduleElement.select("td").eq(0).text());
    }

    private String parseDate() {
        return moduleElement.select("td").eq(1).text();
    }

    private int parseScore() {
        try {
            return Integer.parseInt(moduleElement.select("td").eq(2).text());
        } catch (NumberFormatException ex) {
```

```
            return 0;
        }
    }
}
```

Лістинг коду класу SubjectParcer.java

```
package main.java.api.parser;

import main.java.api.parser.pojo.Semester;
import main.java.api.parser.pojo.Subject;
import org.jsoup.nodes.Element;

import java.util.Collections;

public class SemesterParser {

    private SubjectsParser subjectsParser;

    public SemesterParser(Element semesterElement) {
        subjectsParser = new SubjectsParser(semesterElement);
    }

    public Semester parseSemester() {
        Semester semester = new Semester();
        if (subjectsParser.isEmptySemester()) {
            semester.setSubjects(Collections.<Subject>emptyList());
        } else {
            semester.setSubjects(subjectsParser.parseSubjects());
        }
        return semester;
    }
}
```

Лістинг коду класу SubjectParcer.java

```
package main.java.api.parser;

import org.jsoup.nodes.Element;
```

```
public class NameGroupParser {

    private Element nameGroupElement;

    public NameGroupParser(Element nameGroupElement) {
        this.nameGroupElement = nameGroupElement;
    }

    public String parseName() {
        String rawName = nameGroupElement.text().split("\\(")[0];
        String formattedName = capitalizeString(rawName);
        if (formattedName.length() == 0) {
            return "Ім’я Прізвище";
        } else {
            return formattedName;
        }
    }

    public String parseGroup() {
        try {
            return nameGroupElement.text().split("\\(")[1].split("\\)")[0];
        } catch (Exception e) {
            e.printStackTrace();
            return "ГРУПА";
        }
    }

    private String capitalizeString(String string) {
        char[] chars = string.toLowerCase().toCharArray();
        boolean found = false;
        for (int i = 0; i < chars.length; i++) {
            if (!found && Character.isLetter(chars[i])) {
                chars[i] = Character.toUpperCase(chars[i]);
                found = true;
            } else if (Character.isWhitespace(chars[i]) || chars[i] == '.' ||
chars[i] == '\'') {
                found = false;
```

```
            }
        }
        return String.valueOf(chars);
    }
}
```

ДОДАТОК Л

Звіт про тестування програмної системи

Зміст

1. Вступ

Тестування програмного забезпечення – це оцінка якості розробленого програмного продукту. Якість системи характеризується коректністю, повнотою, безпечністю програмної системи. після виявлення та виправлення кожної помилки слід обов’язково повторити тест.

2. Розробка тестів

2.1 Функціональне тестування – виявлення невідповідностей між реальною поведінкою реалізованих функцій і очікуваною поведінкою відповідно до специфікації і вимог. Функціональні тести повинні охоплювати всі реалізовані функції з урахуванням найбільш ймовірних типів помилок. Тестові сценарії, що поєднують окремі тести, орієнтовані на перевірку якості розв'язку функціональних задач.

На етапі розробки програмної системи було реалізовано 26 функціональних тестових випадків. Таблиця показує результат виконання тестування для кожного варіанта використання.

N/n	Варіант використання	Результат тесту
1.	Авторизація користувача в системі	Пройдено
2.	Зміна семестру	Пройдено
3.	Гортання таблиці	Пройдено
4.	Оновлення Таблиці	Пройдено
5.	Вихід	Пройдено
6.	Збереження паролю	Пройдено
7.	Відкриття сторінки розробника	Пройдено
8.	Закриття сторінки розробника	Пройдено

Функціональне тестування пройшли 8 із 8 тестових випадків, що зображені у таблиці.

Специфікація тестових випадків

N/n	Тестовий випадок	Результат проходження тесту
1.	Перевірка кнопок	Пройдено
2.	Перевірка прокрутки таблиць	Пройдено
3.	Перевірка заповнення обов’язкових полів	Пройдено
4.	Перевірка максимальної довжини полів	Пройдено

Продовження “Таблиці Специфікація тестових випадків”

5.	Перевірка виведення повідомлень підтвердження	Пройдено
6.	Перевірка відповідності введених даних у полях	Пройдено
7.	Перевірка веб переглядача	Пройдено
8.	Перевірка анімацій	Пройдено
9.	Перевірка звукових ефектів	Пройдено

2.2 Тестування безпеки

Для тестування безпеки було розроблено 3 тестових випадки. Всі випадки пройшли тестування успішно. Програмна система є надійною щодо несанкціонованого доступу до даних.

Тестові випадки тестування безпеки

N/n	Тестовий випадок	Результат тесту	Тестові дані
1.	Ввід логіну (менше 4 символів)	Пройдено	Vas
2.	Ввід паролю (менше 4 символів)	Пройдено	123
3.	Ввід логіну кирилицею	Пройдено	Вася
4.	Ввід паролю кирилицею	Пройдено	Пароль
5.	Ввід логіну із спеціальним символом	Пройдено	Vasya$$%

Продовження Таблиці “Тестові випадки тестування безпеки”

6.	Ввід паролю із спеціальним символом	Пройдено	Password$%^
7.	Вхід в систему із вимкненою мережею	Пройдено	-

2.3 Тестування інтерфейсу користувача (GUI). Для тестування було створено 5 тестових випадки для інтерфейсу користувача. Результат проходження тестування зображено у таблиці нижче.

Тестування інтерфейсу користувача (GUI).

N/n	Тестовий випадок	Результат тесту
1.	Перевірка розміщення елементів управління на екранних формах	Пройдено
2.	Перевірка змісту і оформлень повідомлень, що виводяться	Пройдено
3.	Перевірка до форматів введення	Пройдено
4.	Перевірка реакції системи на дії користувача	Пройдено
5.	Перевірка часу відгуку на команди	Пройдено